

689

\$2.95^{USA}

Australia
Singapore
Malaysia

A \$ 4.75
S \$ 9.45
M \$ 9.45

New Zealand
Hong Kong
Sweden

NZ \$ 6.50
H \$23.50
30.-SEK

MICRO JOURNAL

VOLUME VI ISSUE I • Devoted to the 68XX User • January 1984
"Small Computers Doing Big Things"

SERVING THE 68XX USER WORLDWIDE

MC6809E CPU

1	40	41	42
2	43	44	45
3	46	47	48
4	49	50	51
5	52	53	54
6	55	56	57
7	58	59	60
8	61	62	63
9	64	65	66
10	67	68	69
11	70	71	72
12	73	74	75
13	76	77	78
14	79	80	81
15	82	83	84
16	85	86	87
17	88	89	90
18	91	92	93
19	94	95	96
20	97	98	99
21	100	101	102
22	103	104	105
23	106	107	108
24	109	110	111
25	112	113	114
26	115	116	117
27	118	119	120
28	121	122	123
29	124	125	126
30	127	128	129
31	130	131	132
32	133	134	135
33	136	137	138
34	139	140	141
35	142	143	144
36	145	146	147
37	148	149	150
38	151	152	153
39	154	155	156
40	157	158	159
41	160	161	162
42	163	164	165
43	166	167	168
44	169	170	171
45	172	173	174
46	175	176	177
47	178	179	180
48	181	182	183
49	184	185	186
50	187	188	189
51	190	191	192
52	193	194	195
53	196	197	198
54	199	200	201
55	202	203	204
56	205	206	207
57	208	209	210
58	211	212	213
59	214	215	216
60	217	218	219
61	220	221	222
62	223	224	225
63	226	227	228
64	229	230	231
65	232	233	234
66	235	236	237
67	238	239	240
68	241	242	243
69	244	245	246
70	247	248	249
71	250	251	252
72	253	254	255
73	256	257	258
74	259	260	261
75	262	263	264
76	265	266	267
77	268	269	270
78	271	272	273
79	274	275	276
80	277	278	279
81	280	281	282
82	283	284	285
83	286	287	288
84	289	290	291
85	292	293	294
86	295	296	297
87	298	299	300
88	301	302	303
89	304	305	306
90	307	308	309
91	310	311	312
92	313	314	315
93	316	317	318
94	319	320	321
95	322	323	324
96	325	326	327
97	328	329	330
98	331	332	333
99	334	335	336
100	337	338	339
101	340	341	342
102	343	344	345
103	346	347	348
104	349	350	351
105	352	353	354
106	355	356	357
107	358	359	360
108	361	362	363
109	364	365	366
110	367	368	369
111	370	371	372
112	373	374	375
113	376	377	378
114	379	380	381
115	382	383	384
116	385	386	387
117	388	389	390
118	391	392	393
119	394	395	396
120	397	398	399
121	400	401	402
122	403	404	405
123	406	407	408
124	409	410	411
125	412	413	414
126	415	416	417
127	418	419	420
128	421	422	423
129	424	425	426
130	427	428	429
131	430	431	432
132	433	434	435
133	436	437	438
134	439	440	441
135	442	443	444
136	445	446	447
137	448	449	450
138	451	452	453
139	454	455	456
140	457	458	459
141	460	461	462
142	463	464	465
143	466	467	468
144	469	470	471
145	472	473	474
146	475	476	477
147	478	479	480
148	481	482	483
149	484	485	486
150	487	488	489
151	490	491	492
152	493	494	495
153	496	497	498
154	499	500	501
155	502	503	504
156	505	506	507
157	508	509	510
158	511	512	513
159	514	515	516
160	517	518	519
161	520	521	522
162	523	524	525
163	526	527	528
164	529	530	531
165	532	533	534
166	535	536	537
167	538	539	540
168	541	542	543
169	544	545	546
170	547	548	549
171	550	551	552
172	553	554	555
173	556	557	558
174	559	560	561
175	562	563	564
176	565	566	567
177	568	569	570
178	571	572	573
179	574	575	576
180	577	578	579
181	580	581	582
182	583	584	585
183	586	587	588
184	589	590	591
185	592	593	594
186	595	596	597
187	598	599	600
188	601	602	603
189	604	605	606
190	607	608	609
191	610	611	612
192	613	614	615
193	616	617	618
194	619	620	621
195	622	623	624
196	625	626	627
197	628	629	630
198	631	632	633
199	634	635	636
200	637	638	639
201	640	641	642
202	643	644	645
203	646	647	648
204	649	650	651
205	652	653	654
206	655	656	657
207	658	659	660
208	661	662	663
209	664	665	666
210	667	668	669
211	670	671	672
212	673	674	675
213	676	677	678
214	679	680	681
215	682	683	684
216	685	686	687
217	688	689	690
218	691	692	693
219	694	695	696
220	697	698	699
221	700	701	702
222	703	704	705
223	706	707	708
224	709	710	711
225	712	713	714
226	715	716	717
227	718	719	720
228	721	722	723
229	724	725	726
230	727	728	729
231	730	731	732
232	733	734	735
233	736	737	738
234	739	740	741
235	742	743	744
236	745	746	747
237	748	749	750
238	751	752	753
239	754	755	756
240	757	758	759
241	760	761	762
242	763	764	765
243	766	767	768
244	769	770	771
245	772	773	774
246	775	776	777
247	778	779	780
248	781	782	783
249	784	785	786
250	787	788	789
251	790	791	792
252	793	794	795
253	796	797	798
254	799	800	801
255	802	803	804
256	805	806	807
257	808	809	810
258	811	812	813
259	814	815	816
260	817	818	819
261	820	821	822
262	823	824	825
263	826	827	828
264	829	830	831
265	832	833	834
266	835	836	837
267	838	839	840
268	841	842	843
269	844	845	846
270	847	848	849
271	850	851	852
272	853	854	855
273	856	857	858
274	859	860	861
275	862	863	864
276	865	866	867
277	868	869	870
278	871	872	873
279	874	875	876
280	877	878	879
281	880	881	882
282	883	884	885
283	886	887	888
284	889	890	891
285	892	893	894
286	895	896	897
287	898	899	900
288	901	902	903
289	904	905	906
290	907	908	909
291	910	911	912
292	913	914	915
293	916	917	918
294	919	920	921
295	922	923	924
296	925	926	927
297	928	929	930
298	931	932	933
299	934	935	936
300	937	938	939
301	940	941	942
302	943	944	945
303	946	947	948
304	949	950	951
305	952	953	954
306	955	956	957
307	958	959	960
308	961	962	963
309	964	965	966
310	967	968	969
311	970	971	972
312	973	974	975
313	976	977	978
314	979	980	981
315	982	983	984
316	985	986	987
317	988	989	990
318	991	992	993
319	994	995	996
320	997	998	999
321	1000	1001	1002
322	1003	1004	1005
323	1006	1007	1008
324	1009	1010	1011
325	1012	1013	1014



FREE FROM LIMITATIONS. A TEACHER CAN UNLOCK IMAGINATIONS.

By 1985, fully 75% of all jobs will involve computers. So for today's teachers, selecting the right computer is an awesome responsibility. It's a decision you can feel confident about with Southwest Technical Products Corporation's S+ computer system. The S+ is specially designed for a student's tomorrow and a teacher's today. It's built tough. It's simple to operate. You can teach without becoming a hardware specialist.

The S+ is fluent in a variety of languages: BASIC, PASCAL, COBOL, FORTRAN, PILOT,

FORTH, "C," and MUMPS and allows you simultaneous use in the classroom. This flexibility enables you to use your own curriculum, not a machine's version. The system also allows for individual learning styles. Students go as fast and as far as the teacher chooses.


Even though it's priced like a smaller personal computer, the S+ has the multi-task power and expansion abilities of a professional computer. It can grow with you and your students.

Tomorrow is in your hands.

today. Call Fisher Scientific, exclusive distributor of our S+ system, and you'll get the key to unlock imaginations.

For more information.
Call toll-free 800-621-4769.
Illinois customers call collect
(312) 378-7770.

Fisher Scientific

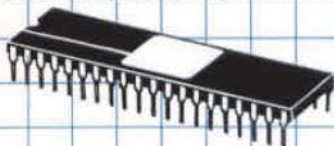
An  **ALLIED** Company

Educational Materials Division
4901 W. LeMoyne St.,
Chicago, Illinois 60651

MICROWARE'S OS-9 IS NUMBER ONE.

OS-9 NOW HAS THE LARGEST USER COMMUNITY

More users now run OS-9 on their 6809 computers than all other operating systems combined. This outstanding success story was no accident — it's due to OS-9's technical excellence backed up by outstanding Microware support. OS-9's Unix-type architecture and totally modular design gives your computer more power and versatility. OS-9 also gives you more possibilities for customization so you can tailor your system exactly to your needs. And aren't flexibility and performance the reasons you chose a 6809 computer to begin with?



OS-9 HAS BEEN CHOSEN BY OVER 50 6809 SYSTEM MANUFACTURERS

OS-9 is now offered as a standard operating system by almost every 6809 system manufacturer, and has been designed into an amazing variety of dedicated systems and products including personal and business computers, process control systems, data and telecommunications systems, and more. In all, over 50 companies and organizations have

obtained OS-9 distribution licenses including such well-known names such as General Motors, NASA, Fujitsu, Western Electric, Motorola, Sykes Datatronics, Eastman Kodak, Thomson-CSF, and Tandy Corp.

OS-9 GIVES YOU A SOFTWARE BASE TO BUILD ON

Whatever your application, OS-9 speaks your language! Microware offers BASIC09, an Extended/Structured Basic, a complete C Compiler, a full ISO Pascal Compiler, the ANSI Standard CIS Cobol Compiler, plus Relocatable Macro Assembler. These high performance programming languages are all fully implemented and deliver unmatched performance and outstanding features. Additionally, OS-9 compatible applications packages such as word processors, screen editors, spreadsheets, business software, and utilities are offered by a rapidly growing number of third-party software houses.

PLUS OUTSTANDING MICROWARE SUPPORT: WE KEEP IN TOUCH WITH YOU

Even when you have the best software and documentation, there can be times when you need questions answered. That's why Microware is committed to giving OS-9 users the best possible personalized service. Here are some

of the ways we deliver solid support:

- A Software Support Hotline for direct access to our technical staff
- "Pipelines", our free quarterly newsletter
- OS-9 User Seminars, the annual OS-9 community gathering
- a liberal update policy for new releases

Microware does business on a person-to-person basis. When you call you'll find yourself speaking with someone who's both knowledgeable and genuinely interested in helping.

YOU CAN COUNT ON OS-9 NOW AND IN THE FUTURE

Microware is not standing still — we're firmly committed to continuing support for the 6809 and we will continue to introduce exciting new software products for the 6809. We will soon announce OS-9/68000 and programming languages for the 68000 which will be upward compatible with 6809 versions, so if and when you are ready for the 68000 your OS-9 software can go with you.



MICROWARE

Microware Systems Corporation
Box 4865 • Des Moines, IA 50304
515-279-8844 Telex 910-520-2535

OS-9 and Basic09 are trademarks of Microware and Motorola.
Unix is a trademark of Bell Labs.

OS-9: BETTER BY DESIGN

'68'

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE

Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-COSI-8212W-Sprint 3 Printer

GIMIX Inc.

1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS

Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

Editorial Staff

Don Williams Sr.	Publisher
Larry E. Williams	Executive Editor
Tom E. Williams	Production Editor
Robert (Bob) Nay	Color Editor

Administrative Staff

Mary Robertson	Office Manager
Joyce Williams	Accounting
Carolyn Williams	Subscriptions
Penny Williams	File Management

Contributing Editors

Ron Anderson
Norm Conno
Peter Dibble
Dr. Theo Elbert
William E. Fisher
Dr. E.M. Pass

Special Technical Projects

Clay Abrams K6AEP
Tom Hunt

CONTENTS

Vol.VI, Issue 1

JAN '84

FLEX USER NOTES.....	10	Anderson
OS9 USER NOTES.....	11	Dibble
A BASIC09 Financial		
Modeling Program.....	17	Bovet
EPSON HX-20 Review.....	23	Stark
ACORN MOTHER BOARD Review.....	25	Abrahams
CCSM Standard MUMPS Review.....	26	Dibble
Converting FLEX to OS9.....	28	Pass
BIT BUCKET.....	44	
HELP.....	47	
CLASSIFIED ADVERTISING.....	47	

MICRO JOURNAL

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

Copyrighted 1983 by Computer Publishing, Inc. (CPI)

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES

USA

1-Year \$24.50 2-Years \$42.50 3-Years \$64.50

FOREIGN

See Page 52

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors full name, address, date and telephone number. It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands ONLY should be used (due to our proportional processor): .sp space, .pp paragraph, .fl fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continuous text.

All articles submitted on diskettes should be in TSC FLEX" format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). Please use a dark ribbon!

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less \$7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

TO REALIZE THE FULL POWER & PERFORMANCE OF THE 6809, LOOK TO GIMIX.

GIMIX OFFERS YOU A VARIETY OF SS50 BUS COMPONENTS AND SYSTEMS.

OS-9 GMX III

The **GIMIX 6809 CPU III** and **OS-9 GMX III**. A Multi-user, Multi-tasking package for the ultimate in System Performance plus protection of the system and other users from crashes caused by errors in individual users programs.

#01 (CPU & Software) **NEW!** \$1698.01

INTELLIGENT I/O PROCESSOR BOARDS increase system throughput by reducing interrupts to the host, buffering data transfers, and data preprocessing. Prices include on board firmware. Requires system drivers.

#11 3 port RS232 Serial (SS30) **NEW!** \$498.11

#12 4 port Parallel (SS50) \$538.12

OS-9 GMXIII drivers (included when purchased with GMX III package) \$200.00

OS-9 Level 2 users - contact GIMIX for system requirements and availability.

192K GMX III #79 SYSTEMS: All include GIMIX 6809 CPU III and OS-9 GMX III (#01); a #11 3 port Intelligent serial I/O & cables; #19 Classy Chassis; 192KB Static RAM; #68 DMA controller, all necessary cables, power regulators, and filter plates. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are included.

#79 with dual 40 track DSDD drives **NEW!** \$5998.79

#79 with dual 80 track DSDD drives \$6298.79

#79 with #88 8" Dual Drive Disk System \$7598.79

#79 with #90 19MB Winchester subsystem & one 80 track DSDD drive \$8998.79

UniFLEX for the GIMIX 6809 CPU III and Intelligent I/O boards is in development.

OS-9 GMX I; OS-9 GMX II; FLEX; and UniFLEX

The **#05 GIMIX 6809 PLUS CPU board** \$578.05

Options: **GIMX DAT** \$35.00 **SWTPC DAT** \$15.00

9511A \$312.00 **9512** \$265.00

#49 64KBHOST SYSTEM Includes: #05 CPU; #19 Classy Chassis; 64KB static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filter plates; GIMIXBUG monitor; FLEX; and OS-9 GMX I. You can software select either FLEX or OS-9. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are also included.

#49 with dual 40 track DSDD drives \$4398.49

#49 with dual 80 track DSDD drives \$4698.49

#49 with #88 8" Dual Drive Disk System \$5998.49

#49 with #90 19MB Winchester subsystems & one 80 track DSDD drive \$7398.49

#39 128KB SYSTEM Includes: #05 CPU; #19 Classy Chassis; 128KB of static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filter plates; GIMIXBUG monitor; FLEX; and OS-9 GMX II. You can software select either FLEX or OS-9. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB, and GIMX-VDISK for FLEX are included.

#39 with dual 40 track DSDD drives \$4998.39

#39 with dual 80 track DSDD drives \$5298.39

#39 with #88 8" Dual Drive Disk System \$6598.39

#39 with #90 19MB Winchester subsystem & one 80 track DSDD drive \$7998.39

UniFLEX, available at extra cost, requires 8" or Winchester drives. A signed license agreement with TSC is required before shipment.

You can add to any GIMIX system RAM, I/Os and other options, or substitute non-volatile RAM. GIMIX will customize to your needs.

COMING SOON: Contact GIMIX for price and availability on 40MB and 72MB Winchester (5 1/4") drives, removable pack Winchesters, 256KB static RAM boards.

All GIMIX systems are guaranteed for 2MHz operation. GIMIX systems include documentation for all boards and software in a GIMIX binder. **ALL DRIVES ARE 100% TESTED AND ALIGNED BY GIMIX.**

ALL BOARDS AND SYSTEMS ARE ASSEMBLED, BURNED-IN, AND TESTED. GOLD-PLATED BUS CONNECTORS ARE USED.

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add \$5 handling if order is under \$200.00. Foreign orders add \$10 handling if order is under \$200.00. Foreign orders over \$200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60603, account #73-32033. Visa or Master Charge also accepted.

EXPORT MODELS: ADD \$30 FOR 50HZ. POWER SUPPLIES.

GIMIX Inc. reserves the right to change pricing, terms, and product specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Choose from GIMIX' wide variety of system components.

The **GIMIX CLASSY CHASSIS #19** consists of a heavyweight aluminum cabinet, constant voltage ferro-resonant power supply, and SS50 Mother board with baud rate generator board

Triple Disk regulator card and cables \$88.22 **Baud rate generator card** \$88.93

Missing cycle detector \$38.23 **Filter plates** \$14.92

Back panel connector plates (specify) \$8.60 **50 Hz. option** \$30.00

MEMORIES (GIMIX uses only Static RAM)

#67 64KB NMOS STATIC RAM board \$478.67

#64 64KB CMOS STATIC RAM board w/battery back-up \$568.64

#34 8K PROM board \$98.34

#32 16 socket PROM/ROM/RAM board \$238.32

I/O Boards (see above for Intelligent I/Os)

#41 Single port serial, RS232/20ma. current loop \$88.41

#43 2 port serial, RS232 \$128.43

#45 8 port serial, RS232 \$318.46

#42 2 port parallel \$88.42

#45 8 port parallel \$198.45

#50 serial, RS232, RS422, RS423 \$244.50

#52 SSDA serial, RS232, RS422, RS423 \$254.52

#54 ADI.C serial, RS232, RS422, RS423 \$268.54

Each cable with connectors for back panel mounting (specify board) \$24.95

DISK CONTROLLERS

#68 DMA (featured in all systems above) \$588.68

#28 dbl. dens. programmed I/O (5" drives only) \$298.28

#58 single dens. programmed I/O (5" and/or 8" drives) \$826.58

#48 same as #58 but for 5" drives only \$198.48

Cable sets: 8" with Back Panel connector \$29.25

for two 8" external drives \$44.26

for two 5" drives \$34.96

SOFTWARE: GIMIX exclusive versions of OS-9/GMX I, II, III & FLEX are for GIMIX hardware only. All versions of OS-9 require the #68 controller.

When ordered with any controller, FLEX is \$30.00

GIMIXBUG PROMS and manual \$98.65

Boot or Video boot PROM \$30.00 **UNIFLEX boot PROM** \$50.00

OS-9 GMX I \$200.00 **OS-9 GMX II** \$500.00

Editor \$125.00 **Assembler** \$125.00

BASIC-09 \$200.00 **RUNB** \$100.00

DISK DRIVES FOR GIMIX SYSTEMS - complete with cables and power regulators.

5" DSDD 40 track 2 for \$900.00

5" DSDD 80 track 2 for \$1300.00

#88" Dual 5" DSDD drives, cabinet, power supply, & cables \$2698.88

Cabinet only \$848.18 **220V 50Hz. Option, add** \$30.00

Filter plate \$14.83 **Cable for 2 drives** \$44.82

Cable for 4 drives \$67.84 **Cable for cabinet to mainframe** \$45.81

WINCHESTER SUBSYSTEMS: for use only in GIMIX systems with #68

DMA controller.

#90: Includes one 19MB drive, interface, and Software \$3588.90

#91: includes two 19MB drives, interface and Software \$5288.91

Contact GIMIX for price and availability of other forthcoming subsystems.

OTHER BOARDS

#76 GHOST 80X24 VIDEO BOARD \$398.76

#66 50 pin Protoboards \$56.86 **#33 30 pin Protoboards** \$38.33

#03 6800 CPU \$224.03

#06 6800 CPU with timers \$288.06 **Baud rate option, add** \$30.00

#08 RELAY DRIVER (board, bracket, transformer, and 31 relays) \$1128.08

#86 - #08 (board, bracket, transformer, without relays) \$538.86

#85 OPTO board \$348.85

WINORUSH EPROM PROGRAMMER \$375.00

3" Binder 12.00 **2" Binder** \$9.00

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS' PRODUCT.

DON'T SEE IT??? ASK! OUR BROCHURE HAS MORE COMPLETE DESCRIPTIONS AND SPECS. PHONE OR WRITE TODAY FOR YOUR COPY.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GIMIX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX INC.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055



Microware presents 4 new OS-9 software packages.

1 LEVEL II PRINT SPOOLING SYSTEM

This versatile package gives your OS-9 Level Two System a complete print spooling management capability for time-sharing applications. Features of the spooling system are:

- Handles up to seven independent spooling devices and queues with "print on first available device" feature.
- Prints large block header pages between listings with date, time, user name and job name.
- Multiple listing copy option.
- Complete forms change capability for each job and device.
- Prints formatted or unformatted listings.
- Status command displays print queues and status.
- User can kill or change priority of queued jobs.

Available only for OS-9 Level Two Systems.

Suggested List Price: \$150.00 Manual Only: \$15.00

2 RMA RELOCATABLE MACRO ASSEMBLER

At last — a full feature relocatable macro assembler and linkage editor for OS-9. RMA permits sections of assembly language programs to be independently assembled to "relocatable object files". The linkage editor takes any number of program sections and/or library sections and combines them into a single executable OS-9 memory module. Global data (including indexed and direct addressing modes) and program references are automatically resolved in the process. The macro facility permits commonly used statement sequences to be defined, then used within the program with appropriate parameter substitution. RMA also supports conditional assembly and library source files.

Suggested List Price: \$200.00 Manual Only: \$20.00

3 OS-9 FILE HANDLER TOOLBOX

Introducing a special toolbox for OS-9 users who do a lot of file manipulation! A collection of 12 useful OS-9 command

programs; Most can be used as "filters" using OS-9 pipeline facilities. Included are:

D — unformatted directory listing with "wild card" matching
Compress — does character compression on text files.
Expand — restores a "compressed" file to the original state.
Split — breaks a file into smaller files.
Space — indents lines with optional spacing between lines.
Code — decodes any key on a keyboard to hex.
Qsort — quick sort for small files, directories, etc.
Tr — transliterates text pattern to substitution pattern.
Grep — searches file for a pattern and prints matching lines.
Xmode — same "tmode" except changes are made to the device descriptor.
Count — counts words, lines, or characters within a text file.

Suggested List Price \$85.00

4 ENTERTAINMENT PACK I

A collection of games and other interesting programs that are not only entertaining but serve as good instructional examples of Basic09 programming techniques. All programs include complete Basic09 source files and can be easily edited to run on standard alphanumeric or graphics terminals.

Blkjak — A Vegas-rules blackjack game.
Cik — graphical display of a wall clock on your terminal.
Dogs — Greyhound racing with simulated graphics.
Eltza — Basic09 version of the famous artificial intelligence simulation of natural language dialogue with a psychiatrist.
Haiku — Program that creates original "haiku" prose.
Quest — a mini-"Adventure" game.
Rats — find your way out of a computer-generated maze — from a rat's point of view.
Towers — a graphical display of the solution to the "Tower of Hanoi" puzzle.

Suggested List Price: \$85.00



MICROWARE.

Microware Systems Corporation
P.O. Box 4865 • Des Moines, IA 50304
515-279-8844 • Telex 910-520-2535

OS-9 and Basic09 are trademarks of Microware and Motorola.
Unix is a trademark of Bell Labs.

FLEX™ USER NOTES THE 6800-6809 BOOK

By: Ronald W. Anderson

As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" \$12.95 — 8" \$16.95 plus \$2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" \$17.95 — 8" \$19.95 plus \$2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All **TEXT** files in the book are on the disks.

LOGO.C1
MOVE.C1
DUMP.C1
SUBTEST.C1
TERM.C2
M.C2
PRINT.C3
MODEM.C2
SCIPKG.C1
U.C4
PRINT.C4
SET.C5
SETBAS1.C5

File load program to offset memory — ASM PIC
Memory move program — ASM PIC
Printer dump program — uses LOGO — ASM PIC
Simulation of 6800 code to 6809, show differences — ASM
Modem input to disk (or other port input to disk) — ASM
Output a file to modem (or another port) — ASM
Parallel (enhanced) printer driver — ASM
TTL output to CRT and modem (or other port) — ASM
Scientific math routines — PASCAL
Mini-monitor, disk resident, many useful functions — ASM
Parallel printer driver, without PFLAG — ASM
Set printer modes — ASM
Set printer modes — A-BASIC
(And many more)

Over 30 **TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1, .C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

Foreign Orders Add \$4.50 S/H

Softcover — Large Format

Book only: **\$7.95** + \$2.50 S/H

With disk: 5" **\$20.90** + \$2.50 S/H

With disk: 8" **\$22.90** + \$2.50 S/H

See your local \$50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

*FLEX is a trademark of Technical Systems Consultants



WE CAN EMULATE THIS CHIP BETTER THAN ITS MAKER.

Advanced Digital's Test Station Integrates and Debugs Your System Faster.

Introducing Advanced Digital's new 4009B Test Station. The test station that combines emulation and logic analysis into one system. Until now, most emulators operated in real-time by restraining the target system. Not with Advanced Digital. Our 4009B Test Station is totally transparent. Totally real-time. We meet or exceed manufacturer's timing specifications. This is particularly important when emulating interrupt intensive systems and accommodating DMA operation.

Universal operation enables you to emulate 6809 and 6809E series microprocessors without making software or hardware changes. Up to 128K of memory overlay, four hardware triggers and 2K deep of trace help you pinpoint problems faster. A line-by-line assembler and a

powerful trace disassembler are standard. They enable you to view software and make changes without returning to the development system.

MORE THAN AN EMULATOR

As a "digital engineering test station," Advanced Digital's 4009B gives you a synergistic approach to integration problems. Two microprocessors control the system. One is dedicated to emulation while the second microprocessor supervises the 32 channel logic analyzer and all system menus. This design lets you emulate and perform all system functions simultaneously.

Discover the most powerful diagnostic tool available for hardware and software integration.

Call Advanced Digital Technology for a free demonstration.



4009B TEST STATION



**Advanced
Digital
Technology**

13400 Northup Way, #27
Bellevue, WA 98005
(206) 643-2382

TMP (Total Management Planning) OS-9 SOFTWARE

THESE OFF-THE-SHELF APPLICATION PACKAGES HAVE CHANGED THE PICTURE
FOR PROFESSIONAL SOFTWARE USERS

Here are the comprehensively supported OS-9 application software packages that discerning professionals are selecting today. Each one runs stand-alone or in combination with other TMP packages.

TMP/CALC is a new generation "Electronic Spreadsheet" package that goes a quantum leap beyond ordinary Calc software. Superior speed, and extra features like Dynamic Calculations and Dynamic Overlay, mean you can change a cell anywhere on the spreadsheet and get automatic updates wherever they belong. No more "wrap-around" since output to the printer is formatted. "Help" screens prompt without erasing data, and the rows and columns on any worksheet are limited only by available memory.
ONLY \$250

TMP/Manager is a structured database manager, and is the ultimate driving force behind any total Management/Marketing Planning system. Its built-in select/sort/merge module — and optimized machine code — makes it superfast, and provides unusually nimble data manipulation characteristics. Other outstanding features are detailed prompting, easy identification of data fields, and an excellent security system.
ONLY \$500

TMP FreeForm fills the void between a structured DBMS and word processing. It's an "electronic index card" package with an endless list of viable applications such as parts cataloging, look-up inventories and listings of any kind. Type anything on up to 32,000 electronic "cards" of up to 9 pages each that reside in "file drawers" (databases). Up to 13 key-words can call each page.
ONLY \$150

TMP/Front-End integrates any or all of the packages described above, and integrates TMP with word processors, BASIC and other high-level languages. NO CHARGE with TMP/Manager



TMP PACKAGES ARE EXCEPTIONALLY FRIENDLY. EACH COMES WITH MULTI-MEDIA TRAINING AND OUTSTANDING DOCUMENTATION

AUDIO CASSETTE TRAINING TAPES are provided with each TMP software package. These crisp, professionally produced training aids take a new user

through "hands-on" exercises that instantly build confidence, and minimize reference to manuals.

CLEAR DOCUMENTATION is specially prepared for each TMP package. Remarkably readable because they are conversational in style and logically organized, this series of manuals further assures quick user comfort and productivity with TMP software. Includes sections on how to best utilize the system; integrating user-written software with TMP packages, and other bonus topics.

VIDEOTAPE PROGRAMS are available to TMP Dealers. They dramatically take the viewer into real business environments for a close look at how a TMP software package is utilized. Video programs can be personalized for specific organizations. Programs may be specified for most popular tape formats.

THE BOTTOM-LINE IS THIS: Software is only as effective as its training and documentation. TMP packages are incomparable in both areas.

RUN TMP IN A SMOKE SIGNAL CHIEFTAIN™ COMPUTER TO EXPERIENCE THE KIND OF SPEED AND POWER YOU CAN EASILY AFFORD TODAY



Endurance-Certified Chieftains are consistently among the fastest computers in CPU and I/O speeds according to the widely respected BENCHMARK REPORTS. That, combined with almost legendary reliability, makes Chieftains the logical choice in computers using OS-9 operating systems.

Configurations range from floppy-disk systems to multi-user, multi-tasking Winchester hard disk systems with tape back up; performance approaching mainframes at prices you can live with.

TMP software packages for most OS-9 systems and Chieftain computers are offered exclusively from Smoke Signal. Inquiries from non-Smoke Signal dealers are invited. Call (213) 889-9340.

TMP T.M.



SMOKE SIGNAL
Chieftain Computers

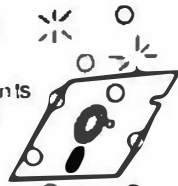
31336 Via Colinas • Westlake Village, CA 91362
(213) 889-9340

BUBBLE MEMORY

HIGH RELIABILITY STORAGE
FOR THE SS-50 BUS

DISK BUB provides 128K bubble memory replacing a disk drive.

- no moving parts to wear out
- direct boot capability
- withstands harsh environments such as dust, heat, vibration
- faster than standard drives
- utility software supplied



Diskbub plugs into the 30 pin I/O bus.
Flex09 drivers and boot rom provided.
Available for only

\$845⁰⁰



2457 Wehrle Drive, B-68
Buffalo, New York 14221
Phone (716) 631-3011

Dealer Inquiries Welcome

Business Software For The COLOR COMPUTER

WITH FLEX* & XBASIC

Data Base Manager

Part I	\$ 49.95
Part II*	\$ 49.95
Church Contributions	\$ 49.95
Balanced Billing System	\$ 49.95
Single Entry General Ledger	\$ 49.95

Integrated Business Software*

*available for Color Computer only

Accounts Receivable	\$ 99.95
Accounts Payable	\$ 99.95
General Ledger	\$189.00
Inventory 2	\$ 69.00
Payroll	\$ 99.95
64K memory upgrade including installation	\$ 79.00



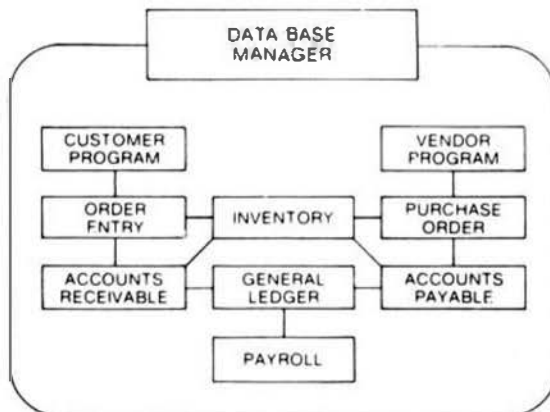
Call or Write for free catalogue

2457 Wehrle Dr., D-80, Buffalo, NY 14221
PHONE (716) 631-3011

Dealer Inquiries Welcome

417

FLEX* and UniFLEX* software for the 6809



Integrated Business Programs

	FLEX	UniFLEX
Accounts Payable	\$295	\$395
Accounts Receivable	\$295	\$395
General Ledger	\$295	\$395
Inventory 2	\$295	\$395
Payroll	\$295	\$395
Data Base Manager	\$350	\$450
Word Processing		
Software	\$295	\$395
WP Menu		\$150
P Control		\$150



*FLEX & UniFLEX are Trademarks of Technical Systems

2457 Wehrle Drive, D-68, Buffalo, NY 14221
PHONE (716) 631-3011

Dealer Inquiries Welcome



Color Micro Journal™

5900 Cassandra Smith Rd. ★

Hixson, TN. 37343

COLOR MICRO JOURNAL™ Is A Monthly
Tabloid Publication for Color Computer USERS!

★★

COLOR MICRO JOURNAL™ is a Magazine FOR
Color Computer Users BY Color Computer Users. Col-
umns on various compatible Operating Systems, Lan-
guages, Uses (Bulletin Boards, Clubs, using the RS
BASIC, and so on), etc.

★★★

Programs - Games - Reviews - Education - Hardware -
Software - New Product Announcements - Books

★★★★

Get the MOST from your COLOR COMPUTER
without being an Engineer.

DON'T MISS A SINGLE ISSUE

Subscription Rate of only \$16.50 a Year!!!

★★★★★

COLOR MICRO JOURNAL™ published by the
ONLY pure 68xx, INTERNATIONAL Computer Maga-
zine. '68 Micro Journal has provided coverage for over
FIVE Years. We KNOW the Color Computer, the Soft-
ware (both FUN and WORK) that IS and CAN BE run on
it. We KNOW the products that ARE, CAN BE, or
WILL BE used on the Color Computer.



Color Micro Journal

Limited Time Charter Rates

USA - \$16.50 per year. Canada & Mexico - \$23.00 per year

Surface Foreign - \$28.00 per year. Airmail Foreign - \$52.00 per year

*Color Micro Journal is a trademark of Computer Publishing Inc.

**For Ordering Subscriptions By Phone
Call 1-800-338-6800**

.....

• Yes! Start my copy of Color Micro Journal coming as soon as possible!

• Name _____

• Address _____

• City _____ State _____ Zip _____

• ☐ Visa ☐ Master Card ☐ Check or Money Order Enclosed

• Card # _____

• Exp/Date _____

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

STANDARDS

I've just received a set of proposed standards for FLEX software from Alan Fowler. He wrote an a letter that appeared in the September (83) Issue regarding the portability (or lack of it) of FLEX software, and how it could be made easier to adapt such software if there were certain standards available to which suppliers would adhere. I find myself in agreement with Alan, most particularly in the area of printer and terminal interfaces. For some reason, there is no standard set of control characters for terminals. While some terminals can do just about anything when given a two character sequence of ESC and a character, some require strings of 5 or 6 characters to do the same thing.

Such problems don't exist if we limit ourselves forever to line editors and non screen oriented software. Most of the "good" software is user interactive, and takes advantage of all of the features of the smart terminal. For example, Reverse Video, Half Intensity, Erase to end of line, Insert line, and Delete line. By taking advantage of these features, a screen menu or a screen oriented editor may be made to run MUCH faster. Our 68XX market, however, is small enough already so that no supplier in his right mind would want to limit his market to only those who have CT-82(XX) terminals, or Televideo, or Soroc... In order to be usable on any terminal, a piece of software needs to be user configurable. The control strings need to be long enough so that even the terminal that requires 4 or 5 characters may be used.

To make things harder, some terminals, including the one on which I am typing at this moment, simply don't have some of these features. I can get along without reverse video and half intensity, by simply outputting nulls to my terminal so that no damage is done. This terminal, however, doesn't have protected fields, so it is not possible to do a "Clear non-protected to spaces" for example. It also cannot do an "Insert line" or a "delete line". On terminals with this feature, it is only necessary to issue an instruction to the terminal to delete a line, or insert one, for example. Without such features, you must rewrite the screen from the cursor to the bottom to duplicate such an action. If a terminal doesn't have Erase to end of line, it is necessary to output spaces from present cursor location to end of line.

I will praise any supplier who takes advantage of these features of the smart terminals, but I would hope that he would make his software smart enough to recognize the fact that the user had configured the control string to all nulls, and on that signal, jump to a routine to "do it the dumb way". For the user, there is the advantage that some pretty fancy software will work on even the dumbest of terminals. For the supplier, there is the advantage that he is not "ruling out" any SS-50 bus system owner from using his software. If a user has a smart terminal, he will get the best possible operation. If he has a dumb one, he will understand that the limitation is in his equipment and not the software. I run my Dumb Terminal (you guessed correctly by now that it is an ADM-3A) at 19.2K baud, and it does reasonably well with software that rewrites the screen every time something fancy is done.

Incidentally, though I have used a number of other terminals, and I must say that I like some of the nice displays that may be done with reverse video and half intensity, I have never found a keyboard that surpasses my old ADM-3. Many of the newer ones could be described as "skritch" (the sound they make). Also, the newer ones seem to "make" about the time they are touched. The old ADM keys don't make until they are about 80% depressed. Perhaps it is just a matter of "what I am accustomed to", but I really think the quality of keyboards has gone down considerably in the past few years. I won't belabor the point further.

Spelling Again

Hey, Don Williams! At the end of my September column you inserted a note that some FLEX versions "encompass enchantments" coded by SWTP programmers. Of course you

meant "enhancements" but a reader wrote to say that he "always wondered why 68xx code always seemed to have such magical qualities." If you are going to add to my efforts, you could at least identify your additions by using "(Ed.)" so I wouldn't get the blame! (And you've been bugging me to use a spelling checker). Seriously, the additions and clarifications are appreciated.

The Other Operating System

Right! I'm going to admit it. FLEX is not the ONLY operating system for our SS-50 (and TRS Color) computers. Right now, I am getting the pieces together to be able to run OS-9. Now I know that may sound like heresy coming from the author of "FLEX User Notes". Why then, am I going to explore OS-9? There are a couple of reasons. First is the chance that I might miss some good software that is available only in OS-9 version. Second, I think there is a place for an operating system that is more advanced than FLEX but doesn't require all the hardware required by UNIFLEX. That is, something sort of "in between". My first impression is that OS-9 might fit that slot, and perhaps be more appropriate for the hobbyist than UNIFLEX. I could go on with further first impressions, having finally gotten myself to read through an OS-9 manual, but I will reserve further comments for AFTER I have had a chance to run some things on my system for a while.

For now, just let me say that there are applications where one is more appropriate than the other. After I've sorted out some impressions I will report here. As my regular readers know, this column has sort of drifted into the area of compilers and software evaluations, though I try to keep a few good applications coming from month to month. I've never been heavily into describing or discussing the inner workings of FLEX, and I don't intend to get into that area of OS-9 (which is being covered very nicely by Peter Dibble in his column). Perhaps eventually, I'll have to rename this column to something like 68XX User Notes!

COMPILER - ASSEMBLER DEBATE

Last November's '68' Micro Journal contained some thoughts from me concerning the relative merits of Compiled languages and Assembler code. I mentioned letting Dan Farnsworth respond to it, and promptly forgot to send him a copy so he could respond and we could continue the discussion on a more timely basis. Dan has responded to my thoughts in that column, and I will quote his thoughts below.

"I would like to divide program creation into two functions. Writing the program and Coding the program. The Writing phase will take up most of the time and should be done in English. You should use diagrams, charts and any other methods for organizing your thoughts in a logical manner. When the Writing phase is completed you can now Code the program into the computer using any language that you have, in a relatively short time. It is my contention that Assembly language will run faster, take up less memory space, is more powerful, and easier to document than any of the High Level languages available.

High Level Languages contain a collection of subroutines which do the actual work. These subroutines are organized by the Syntax or instruction set of the HLL. I freely admit that I cheat when using the Assembler. Over the years I have collected 350 subroutines and burned them into 3K of ROM at the high end of memory. I also have about 50 pointers that are equated to the lower half of page 0. Now by LIB(rary)ing the two equate files I have loaded my new program to a very powerful runtime package. It is easy to Code the program using the Assembler to setup and run the subroutines.

The modern Assemblers which are available will catch 90% of the errors. It only takes a minute or two to run the Assembler and I usually do this every 30 minutes. As each Module is entered I get it to run and Debug it before continuing. The use of Pointers in Page 0 for temporary storage and work space, makes debugging much easier than using the stack. When a program bombs, A listing of page 0 will usually tell you within 5 lines of where the program took off. The use of subroutines that have previously been debugged is also a great help in getting programs to run properly with a minimum of hassle."

Dan's letter continues indicating that he will send me some further thoughts getting into more detail, including some examples of code, etc. He also has invited me to write my "rebuttal" of the above.

Dan, it seems to me that we agree on more points than we disagree. I will make a few points, though. First I agree in principle about the separation of the writing of the program and the coding. When I test a new compiler, I generally try to separate out the "writing" errors from the "coding" errors, by translating a program written in another similar language to the new one. You might say that I am just "re-coding" that program. When I do that and the program doesn't run, I can pretty well assume that some statement in the new language didn't do what I expected it to do. In other words, I made a coding error. (Sometimes I find that the new compiler doesn't do what its author expected, and indicated that it would do in the manual).

Perhaps much of what I do is repetitive so that I pretty well understand the "writing" part of the programming, and I can proceed to the coding. Dan, I think if you were to work in Pascal or PL/9 for a while you would begin to see a GREAT similarity between your program and the English writing of the program you mention in the first paragraph of your discussion. I've tried several times to start writing an article about programming in general (or rather in English), to discuss WHILE DO loops, and REPEAT UNTIL loops, etc. Any discussion of such programming concepts leads immediately to English statements that look so much like Pascal, that I might as well use Pascal for the discussion. I might add that PL/9 is only a tiny bit less "English like" than Pascal, and "C" just a little more abstract, once you understand the syntax.

Let's look at efficiency for a few moments. Suppose, in an assembler program you want to create a variable and then later in the program you want to set it to the value (decimal) 100. In assembler you would create the variable (I'll use a single byte variable) by the statement VARIABLE RMB 1. You would set it equal to 100 with two statements, LDA #100 - STA VARIABLE. In PL/9 you would create the variable with BYTE VARIABLE; and set it equal to 100 with VARIABLE = 100;. If you do just that in PL/9 you will find out that PL/9 generates code just as efficient as that I used in the assembler version above. (Actually, byte variables are handled in PL/9 by the B accumulator, so that the generated instructions would be LDB #100 - STB OFFSET,Y. Just as in the case of the assembler above, if the number of variables is small, they are all accessed with a two byte code. An assembler programmer would probably use direct page for a small number of variables, and STA direct is a two byte code as is STB indexed with a small offset. I agree that this is a simple example, but many of the compiled languages now output code that when disassembled looks "just like someone did it in assembler" a great deal of the time. Although I've used PL/9 in this example, I have more than one of each of Pascal and C compilers that generate code just as efficiently.

Another point that you made is that you had written a number of subroutines that you use in writing programs in assembler. In effect, you have written your own compiler in "secret code" that no one else can understand as easily as you. Using a standard language results in a program that may be understood by someone else quite easily. I realize that there are situations in which no one else will ever have to understand a program, and that is a "so what" situation. However, as programs become larger and larger, and reach the point where more than one person has to work on them, or when they must be maintained for customers over a long period of time, so that there is no guarantee that the original author will still be an employee of the company, (or for that matter, might be ill or even die), the use of a higher level language makes much more sense.

Now having said that, let me indicate an exception. A company recently came to me for some consulting in the area of programming. I started trying to convert them to a HLL immediately. Later I found out that the product (or at least one of them) is a high volume item (sales in the thousands per year) and the program is 2 or 3K of assembler code. Not only do I think Assembler is ideal for this sort of application, I agree that this company should still be using the 6800 rather than the 6809 because of the cost differential. The cost of writing the program in assembler might be as much as a dollar per unit. So what? If the smaller code can save one \$4.00 EPROM the savings are considerable.

One area in which we agree completely, is that programs should be written in small modules, and the modules tested one at a time. In a high level language, each module is a Procedure or Function. You can write a simple test program for each one, or continue to add new procedures and expand the "old program" which becomes the test program for all the procedures. When you are

done adding procedures, your program will be done. Of course, in Assembler programming, the modules are subroutines, and you can go through exactly the same process to get your program running. A common but very costly practice of novice programmers is to write an entire program and then start debugging it. (I know, I used to do it that way myself.) Assembler programmers tend not to make some particular module a subroutine unless it is used more than once in the program. More experienced programmers in both Assembler and HLL's tend to write shorter modules and separate them out of the main program even though they are only used once, since this practice results in shorter modules to debug, and greatly facilitates debugging.

Many BASIC and Assembler programmers are puzzled by the concept of "local variables" in the structured languages. The use of local variables in a procedure of function, greatly reduces the chance of accidentally "clobbering" the value in a variable that was not intended to be modified by the procedure. Good programmers tend to maximize the use of local variables (these only exist while the procedure or subroutine is running). They tend to pass the procedure the information that it needs to calculate its result, and let the procedure return its result in a very controlled manner rather than allowing all parts of the program to access all the variables. There is not enough space here to go into this aspect of programming as much as I would like at this time, but it can be the topic for discussion in a future column. Of course a good Assembler programmer can use the same techniques to minimize interaction of modules, but most do not.

Lastly, I would like to expand on Dan's comment "You should use diagrams, charts, and any other methods for organizing your thoughts in a logical manner." I don't believe in flow charts in general. I've found that the only way to get them to represent a program accurately, is to prepare them after the program is debugged. However, they are VERY useful to straighten out particularly complex logic in a section of code where a number of decision points exist, and the code to be executed in each case is not quite as straightforward as in a "case statement". In such cases, I've resorted to a flowchart with excellent results. By all means use whatever method you know that can help you organize your program.

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

More About Computers at School

I had my first chance to look through a microscope when I was very young. My sister was deeply engrossed in the microscopic world so I, being a typical younger brother, hung around and made a pest of myself until she showed me what she was working on. I couldn't see anything but a blur which sometimes faded out altogether. I didn't see much of it in looking at a blur. As years went by I was given my own microscope, but chemistry sets, and my own experiments, were much more interesting. I still had trouble getting interested in blurs.

In ninth grade I encountered a real microscope for the first time. It was a fine old instrument. The teacher treated it with great respect, and insisted that we do the same. When I first used it I got a surprise that stays with me to this day. It was nothing like the microscopes I had used before, focusing it with the fine adjustment knob was no problem, and when something was in focus, even a single cell or a bacterium, it was very clear. I could have happily spent weeks peering through the eyepiece at everything I could fit on the stage. Eventually the class moved on to other things, but I had a new appreciation for the world of the very small.

It is unfair to blame my parents for not getting me a high quality microscope when I was eight, but it bothers me to think of what I missed. I was fascinated by what the microscope revealed when I was a teenager. The effect would have been even stronger if I had been younger.

My experience with the microscope is what makes me keep complaining about the tendency of schools to use the lowest quality hardware and software they can find.

The younger the students, the lower the quality. The argument is that sophisticated hardware and software isn't needed for any but the most advanced students. This is a serious error. With computers "foolproof" means either trivial, or very sophisticated. It requires good hardware, and excellent software to deal satisfactorily with the worst a child can do. The kids at most schools are getting the same kind of experience with computers I got with my early microscope ... only a blurry image of what it should be.

The section of a column I wrote a few months ago about computers for schools has drawn more comment than any other column I have written, maybe more than all of them put together. Some people wrote to agree, others disagreed. I was glad to hear from those who agreed with me, but I was most interested in the letters from people who took issue with one or more of my points. Two of my points drew particularly heavy criticism. I calculated the price of an imaginary (but realistic) single-user computer. Several people thought an adequate computer could be purchased for less than I suggested. I also spent some time wishing schools would stop using Basic. It didn't surprise me that several readers felt Basic was a fine language.

The little story about the microscope was intended to address the question: "Why bother to provide decent computers at school?" Students should be given a chance to use a computer that they don't have to struggle with, and a language that encourages clear thinking. Kids don't know enough to complain about Basic on the cheapest computer that can be found. I do, so I am complaining for them.

There were about five more paragraphs here about Basic, and the evils of skimping on computers for children, but while re-reading the column I decided that I sounded a bit shrill. Please forgive the abrupt transition, but the smooth conclusion of this argument has been pruned with a quick block-delete.

Pipes

One of the most useful features of OS-9 (and UNIX) is the pipe. Pipes by themselves aren't good for much, but if you build a good set of "software tools," pipes make many tasks surprisingly easy.

A pipe is a special device which forms a connection between two programs such that the output from one is directed into the input of the other. The shell is a major user of pipes. You can ask the shell to connect the standard output of one program to the standard input of another by putting an exclamation point "!" between the commands. The "!" separates commands like the ";" and "&" do, but it also redirects the output of the command before it into the input of the command after it. You could get the same effect by using intermediate files (Have the first command save its output into a disk file. When the first command ends, run the second command with its input coming from the file the first command wrote.), but intermediate files are neither as fast nor as easy to use as pipes.

When you first start using OS-9, pipes won't be of much use to you. For one thing they are a bit confusing, but, more important, the standard OS-9 utilities don't include many filters.

A filter is a program which reads from the standard input file and writes to the standard output file until end of file on standard input. They can be used without pipes, but, in combination with pipes, a good toolbox of filters can be among the most useful facilities available under OS-9.

The most elementary filter would simply copy bytes from standard input to standard output. More advanced filters change data on its way through. Some common filters sort the data, break it into words, remove duplicate lines, count bytes, words, and lines, and translate upper case letters to lower case.

It is relatively easy to write special filters to solve problems one at a time. The trick is to write filters which, in combination with others, can do lots of useful things. I have a filter which I call "words" (available from the OS-9 Users Group, but too long for this column) which breaks the input up into one word per line. I wrote another program which counts the number of <CR>s in the input and writes that number out when the end of the input is reached. I can hook those two programs together with a pipe to form a command line that counts the words in a file:

OS9: words <column10 ! linct
That command line feeds column10 into words which slices it up, one word per line. The output of words is fed into the standard input of linct which responds by giving me the number of lines in its input -- the number of words in column10. I can use linct by itself to find the number of lines in a file.

I have written other filters called sort and uniq. Sort sorts the standard input into the output. Uniq removes duplicate lines; for example:

```
Line One
Junk Line
Junk Line
Junk Line
Another line
would come out of Uniq
Line One
Junk Line
Another line
```

The command line:

OS9: words <column10 ! sort ! uniq
would break column10 into words, sort the list, remove duplicate lines, and give me a sorted list of the words I used in that column.

Since I have written a number of programs in assembler and Basic09 for this column, I thought I might include a few filters written in Pascal this month. Unfortunately old releases of OS-9 had a flaw in PIPEMAN which prevented it from working with Pascal programs. Pascal rewinds its standard input file when it starts. PIPEMAN wouldn't put up with a rewind with the upshot that filters written in Pascal couldn't even get started. The easiest language I know for writing filters is C, but since C isn't as widely used as assembler and Basic09, I'll include two filters, BWord in Basic09, and CharCt in assembler.

Both BWord and LineCt are crude programs. They are nowhere near as efficient as they can be. In particular, reading one character at a time is intensely bad practice under OS-9. Both of these programs could be generalized by using command parameters more extensively.

CharCt counts the number of occurrences of the first character in the command line parameter area in the standard input file. It could be generalized to look for character strings, or regular expressions. It might also be improved by using more than three bytes for the counter.

The shell always places at least a carriage return in the parameter area passed to a program. It starts (FORKS). CharCt relies on this to give it an easy way to default to counting carriage returns in its input. If you want to count some other character use it as a parameter on the command line:

OS9: charct <testfile
would count periods in testfile.

OS9: charct <testfile
would count carriage returns in testfile.

BWord splits the input file into lines, one word per line. A word is defined as a string of characters between spaces, tabs, or carriage returns. It would be more generally useful if it would define a word as a string of characters delimited by any given set of characters. One use of this that comes to mind is to divide a file into sentences by breaking it at each period.

BWords should be entered with Basic09, and packed. If you have RUNB you can run words with a command line like:

OS9: words <testfile
which will divide the text in testfile into words. If you don't have RUNB you might need to use a somewhat longer command line:

OS9: basic09 words <testfile
It is easy to spend a great deal of effort writing filters you will never use. What is needed is a set of general purpose tools. There are several sources for good ideas for filters. Books about UNIX often give descriptions of filters which are commonly used under UNIX. In general, if a concept is useful for UNIX it will also be for OS-9. The standard programming book, Software Tools, by Kernighan and Plauger, is an especially good source for ideas and algorithms.

A More Advanced Approach to Pipes

The Shell uses pipes to connect strings of its children together. Any program that has access to OS-9 system calls can use the same trick the shell uses to make the standard output of one of its children feed directly into the standard input of another, but it is simpler to use pipes as a connection between a process and its parent. If you need a formatted list of processes (the information given by the procs command) you can either mess with the process descriptors yourself, or use a pipe to intercept the output from procs.

If your algorithm can be divided into several sections that communicate in only one direction (Say, one section collects information, the second sorts it, and the third formats a report.), the job can easily be done by three separate processes dispatched from the command line with the shell managing the pipes. If the

steps aren't fixed (Perhaps you either report or update a file depending on the date.). It might be easier to deal with the pipes yourself. This type of thing requires pipes to be defined for each new process's standard input path.

Using a pipe as the standard output path from a child process is useful for more than intercepting the output from system utilities. The first experiments to try with this mechanism are with system utilities, but the most interesting applications are with processes designed especially for this use. An example might be a program which uses a process attached via a pipe to get data from a remote computer. The process at the end of the pipe would dial the remote computer up, go through the login formalities, and deal with any communication protocols. The main process would just read distilled information from the pipe.

All three standard paths can be used for pipes. I haven't thought of a use for all three paths, but a combination of input and output paths is useful. The child process is given work to do through its standard input path and returns the results of its work through its standard, or error, output path. The parent process gives the child work through one pipe and at an appropriate time (maybe much later) gets the results by reading from a different pipe.

A FORKED process inherits the three standard paths of its parent. If it were OK to give up after setting up pipes, the way to set them up would be to close the standard files, and create three pipes, one each for path one, two and three. The instructions to open a pipe in the standard input path would be:

```
Pipe fcs "/PIPE"
lde #0 std in
OS9 $CLOSE
leax Pipe,PCR
lde #UPDATE.
OS9 $OPEN
```

New paths always take the lowest available path number so the pipe would fall into path zero. A process forked from this process would inherit its standard paths including the pipe in path zero. The new process would treat its path 0 as a normal standard input path. Characters written into the pipe by the parent would be read by the child.

If a pipe is opened with no process FORKED to use it, the pipe will act like a queue. A process can write a limited number of bytes into the pipe and read them out again in the same order they were written. If there isn't room in the queue for the data from a write to be stored the process doing the write will be put to sleep until there is space to complete the write. If the process that reads from the pipe is the same one that is sleeping until the queue empties a little there is a deadlock. A deadlock can only be avoided, or broken by some outside agency ... the human at the terminal for instance. Because of this deadlock problem, and the small size of the queue in the pipe, the idea of using a pipe as a queue is only a novelty.

The example of communications via pipes that I have invented is a Basic09 program that prompts for pairs of coordinates, and passes the pairs to a C program which "rasterizes" the lines between the points defined by the coordinates. The Basic09 program passes as many pairs as it likes to the C program, then closes the path it has been writing the data to. When the parent closes his end of the pipe the child will get an end-of-file. The C program sends the rasterized data back through its standard output path. This data consists of a string of zeros and ones indicating where dots should be placed on each horizontal line in order to draw the vectors received as input.

Rasterizing vector graphics information is a particularly good application for a separate process. In a Level Two system each process can use an entire address space of almost 64K. The size and resolution of the graph that is produced depends on the amount of memory available for the bit map of the graph. I have a version of rast that uses 46K for its bit map and can generate an 8"x8" graph on my Okidata at 72 dots per inch. I am not very experienced with graphics; there is probably a much better way to rasterize data than what I used. My program seems too complicated for such a simple task, but it works.

It is particularly important to keep track of interactions between two processes communicating via pipes. If the processes ever get into a situation where both are waiting for input from a pipe leading to the other process, they will be stuck until you free them by killing one of the processes.

The important part of this system of programs is an assembly language subroutine for the Basic09 program. The subroutine is descended from the StrtTask subroutine I published months ago, but has been

enhanced to open pipes to the new process. The \$DUP call is used to preserve the standard input and output files of the Basic09 program while paths zero and one are turned into paths then back into whatever they were before.

Installation

This system of programs is written in three separate languages. If you don't have C it should be fairly easy to translate rast into Basic09, but if you rewrite rast in Basic09 be certain that you don't try to fork it directly. Basic09 should be the program you fork; rast should be the parameter. If you want to keep the old StrtTask around, rename either it or the new one. Grapher should be typed into Basic09 and saved. Particularly if you are using Level One, you should pack Grapher and use RunB to save memory. In Summary:

```
Enter StrtTask and rast.c using an editor
Assemble StrtTask
Compile rast.c
Enter Grapher using Basic09
Save the source
If you intend to run Grapher from the command line
add the line: BYE to the end of Grapher
Pack Grapher
Run Grapher
It will load StrtTask and rast from the execution
directory
```

Operation and Modification

Grapher will prompt for pairs of coordinates. After each pair is entered it will ask you to verify that you want to plot that line. Be careful with this. There is no validation in any part of this system. There is no reason it shouldn't be there either. Please add enough error checking to make you comfortable if you intend to do more than play with this program a little. If you try to draw a line way off into the wild blue yonder your computer will give it a good try, mashing everything in its way. After you enter the last pair of coordinates respond to the (y,n,d) prompt with D. The D response sends the last pair to rast and charts the response from rast on the screen. I like to draw conservative patterns like the one given by:

```
0 0 79 23
0 23 79 0
0 0 0 23
0 0 79 0
0 11 79 11
39 0 39 23
```

Rast is set up to rasterize a 80 by 24 graph. That is the size of a standard terminal, but if you want to deal with larger or smaller graphs, change VDIMENSION to the number of vertical dots in the graph, and HDIMENSION to the number of horizontal dots.

Pipes are a powerful tool for interprocess communications. They can be used with good effect to solve almost any interprocess communication problem if the connection can be made. The worst problem with pipes is that they can only be used between processes that are very closely related (between siblings, or parent/child). There is also a performance problem under Level Two; not only is there the cost of a system request per transfer, but OS-9 has to move the characters from one address space to another -- taking a surprising length of time. If you feel ambitious you will find that it is possible to make a major performance improvement to rast by using a compression algorithm on its output.

Welcome COCO

I have been reading messages in the COCO special interest group on Compuserve. It sounds like Microware put a real version of OS-9 on that little machine. I am seriously impressed with the reality of a very inexpensive computer with a UNIX-like, multitasking, even -- if I may stretch a point -- multi-user operating system. There may be a number of interesting ways to integrate COCOs with each other and with larger OS-9 systems to get a bargain version of advanced distributed computing. It may not be too much to hope for that Tandy will find a way to put OS-9 Level Two on some descendant of the COCO. There is some chance that I will be able to take the viewpoint of a COCO user in this column in the future. I haven't made up my mind yet, but I need a Level One system, and the Color Computer may be the way to get one. I would appreciate advice.

The Users Group

The executive committee of the OS-9 Users Group has met twice since the annual meeting (I am writing this in November). We have struggled with various issues and defined assorted policies, mostly rather dull. Very likely by the time this column is printed the members will have received a newsletter, and everyone will have seen information in this and other magazines. Right now our software library is ready to go. I know it has good stuff in it; several programs of mine are part of the collection. Our plan is to give a standard selection of software from the library to the existing membership and to each new member. The other programs in the library will be available for small amounts of money, or software contributions. The address of the Users Group is:

OS-9 Users Group
PO Box 8027
Des Moines, Iowa 50301

PROCEDURE word

```
0000 10 -----a)
0001 10 Fail to divide input into words. One word per a)
0002 10 line. 4)
0003 10 -----b)
0004 DIM chr:BYTE
0005 DIM inword:BOOLEAN
0006 DIM StdOut, StdIn, StdErr:INTEGER
0007 ON ERROR GOTO 100
0008 StdIn=0
0009 StdOut=1
0010 StdErr=2
0011 inword=FALSE
0012 LOOP
0013 GET StdIn,chr
0014 IF inword THEN
0015 IF chr=ASC(' ') OR chr=9 OR chr=13 THEN
0016 inword=FALSE
0017 WRITE StdOut
0018 ELSE
0019 PRINT StdOut,chr;
0020 ENDIF
0021 ELSE
0022 IF chr=ASC(' ') OR chr=9 OR chr=13 THEN
0023 ELSE
0024 inword=TRUE
0025 PRINT StdOut,chr;
0026 ENDIF
0027 ENDIF
0028 ENDIF
0029 ENDIF
0030 ENDIF
0031 ENDIF
0032 ENDIF
0033 ENDIF
0034 ENDIF
0035 ENDIF
0036 ENDIF
0037 ENDIF
0038 ENDIF
0039 ENDIF
0040 ENDIF
0041 ENDIF
0042 ENDIF
0043 ENDIF
0044 ENDIF
0045 ENDIF
0046 ENDIF
0047 ENDIF
0048 ENDIF
0049 ENDIF
0050 ENDIF
0051 ENDIF
0052 ENDIF
0053 ENDIF
0054 ENDIF
0055 ENDIF
0056 ENDIF
0057 ENDIF
0058 ENDIF
0059 ENDIF
0060 ENDIF
0061 ENDIF
0062 ENDIF
0063 ENDIF
0064 ENDIF
0065 ENDIF
0066 ENDIF
0067 ENDIF
0068 ENDIF
0069 ENDIF
0070 ENDIF
0071 ENDIF
0072 ENDIF
0073 ENDIF
0074 ENDIF
0075 ENDIF
0076 ENDIF
0077 ENDIF
0078 ENDIF
0079 ENDIF
0080 ENDIF
0081 ENDIF
0082 ENDIF
0083 ENDIF
0084 ENDIF
0085 ENDIF
0086 ENDIF
0087 ENDIF
0088 ENDIF
0089 ENDIF
0090 ENDIF
0091 ENDIF
0092 ENDIF
0093 ENDIF
0094 ENDIF
0095 ENDIF
0096 ENDIF
0097 ENDIF
0098 ENDIF
0099 ENDIF
0100 ENDIF
0101 ENDIF
0102 ENDIF
0103 ENDIF
0104 ENDIF
0105 ENDIF
0106 ENDIF
0107 ENDIF
0108 ENDIF
0109 ENDIF
0110 ENDIF
0111 ENDIF
0112 ENDIF
0113 ENDIF
0114 ENDIF
0115 ENDIF
0116 ENDIF
0117 ENDIF
0118 ENDIF
0119 ENDIF
0120 ENDIF
0121 ENDIF
0122 ENDIF
0123 ENDIF
0124 ENDIF
0125 ENDIF
0126 ENDIF
0127 ENDIF
0128 ENDIF
0129 ENDIF
0130 ENDIF
0131 ENDIF
0132 ENDIF
0133 ENDIF
0134 ENDIF
0135 ENDIF
0136 ENDIF
0137 ENDIF
0138 ENDIF
0139 ENDIF
0140 ENDIF
0141 ENDIF
0142 ENDIF
0143 ENDIF
0144 ENDIF
0145 ENDIF
0146 ENDIF
0147 ENDIF
0148 ENDIF
0149 ENDIF
0150 ENDIF
0151 ENDIF
0152 ENDIF
0153 ENDIF
0154 ENDIF
0155 ENDIF
0156 ENDIF
0157 ENDIF
0158 ENDIF
0159 ENDIF
0160 ENDIF
0161 ENDIF
0162 ENDIF
0163 ENDIF
0164 ENDIF
0165 ENDIF
0166 ENDIF
0167 ENDIF
0168 ENDIF
0169 ENDIF
0170 ENDIF
0171 ENDIF
0172 ENDIF
0173 ENDIF
0174 ENDIF
0175 ENDIF
0176 ENDIF
0177 ENDIF
0178 ENDIF
0179 ENDIF
0180 ENDIF
0181 ENDIF
0182 ENDIF
0183 ENDIF
0184 ENDIF
0185 ENDIF
0186 ENDIF
0187 ENDIF
0188 ENDIF
0189 ENDIF
0190 ENDIF
0191 ENDIF
0192 ENDIF
0193 ENDIF
0194 ENDIF
0195 ENDIF
0196 ENDIF
0197 ENDIF
0198 ENDIF
0199 ENDIF
0200 ENDIF
```

```
00001 NAME CharCt
00002 TTL Count a occurrences of a specified character
00003 0-----a)
00004 * CharCt Written 1 November 83
00005 * Last Modified 5 November 83
00006 * A filter to count occurrences of any specified
00007 * character in the standard input. If no
00008 * character is specified, default to counting
00009 * carriage returns.
00010 0-----b)
00011 IFP1
00012 ENDC
00013 00014 0011 Type set Prgr=Objct
00015 0001 Revs set ReEnt=1
00016 0000 07C0000A MOD pgen=CharCt,Type,Revs,Entry,Revs
00017 0 0000 Count rob 3 stored in BCD
00018 0 0003 InChr rob 1
00019 0 0004 TstChr rob 1
00020 0 0005 OutStr rob 6
00021 0 0008 CR rob 1 for a CR
00022 0 000C rob 200 Stack
```

```
00023 0 0004 Revs= equ
00024 0000 43686172 CharCt fcs /CharCt/
00025 0013 01 fcb 1 version
00026 -----
00027 * At entry:
00028 * U and DP point at local storage.
00029 * X points at the parameter area.
00030 -----
00031 0014 Entry
00032 0014 0F00 clr Count
00033 0016 0F01 clr Count+1
00034 0018 0F02 clr Count+2
00035 001A A684 lda ,1
00036 001C 9704 sta TstChr
00037 001E 3043 leax InChr,U
00038 0020 108E0001 ldy 01 characters to read
00039 0024 Loop
00040 0024 8600 lda 00 std in
00041 0026 103F89 OS9 I$READ
00042 0029 251E bcs Quit
00043 002B 3603 ldx InChr
00044 002D 8104 ccmp TstChr
00045 002F 26F3 bne Loop
00046 -----
00047 * Increment Count
00048 -----
00049 0031 8601 lda 01
00050 0033 9802 addx Count+2
00051 0035 19 daa
00052 0036 9702 sta Count+2
00053 0038 8600 lda 00
00054 003A 9901 adca Count+1
00055 003C 19 daa
00056 003D 9701 sta Count+1
00057 003F 8600 lda 00
00058 0041 9900 adca Count
00059 0043 19 daa
00060 0044 9700 sta Count
00061 0046 4F clra std in
00062 0047 200B bra Loop
00063 -----
00064 * If we reached EOF print the total count and
00065 * exit.
00066 * If some other caused us to stop. Return
00067 * with an error code.
00068 -----
00069 0049 Quit
00070 0049 C103 ccmp 0F3EOF
00071 004B 2636 bne Exit
00072 004D 3045 leax OutStr,U
00073 004F 9600 lda Count
00074 0051 8D33 bsr Cnvt
00075 0053 9601 lda Count+1
00076 0055 8D2F bsr Cnvt
00077 0057 9602 lda Count+2
00078 0059 8D2B bsr Cnvt
00079 005B 3045 leax OutStr,U
00080 005D 960A lda OutStr+5 mark last position in OutStr
00081 005F 8A80 ora 080 set carry bit
00082 0061 970A sta OutStr+5
00083 0063 108E0007 ldy 07 length
00084 0067 8630 lda 00
00085 0069 FndLen
00086 0069 A184 ccmp ,1
00087 006B 2606 bne OutPut
00088 006D 313F leay -1,Y decrease length
00089 006F 3001 leax 1,X
00090 0071 20F6 bra FndLen
00091 0073 OutPut
00092 0073 8600 lda 00 <CR>
00093 0075 9708 sta CR
00094 0077 960A lda OutStr+5
00095 0079 847F andx 007F clear the carry bit out
00096 007B 970A sta OutStr+5
00097 -----
00098 007D 8601 lda 01 std out
00099 007F 103FBC OS9 I$WRITE
00100 0082 3F clrb clean up for exit
00101 0083 Exit
00102 0083 103F06 OS9 F$Exit
```

```
00000 errors)
00000 warnings)
$007A 00154 program bytes generated
$00D4 00212 data bytes allocated
```

[illegible]

```

00009      *   Return the new tasks process number, the path
00010      *   numbers for the pipes, and the condition code
00011      *   from the Fort.
00012      *   Calling sequence:
00013      *   run StrtTask (Name, Process_Num, Lang_Type,
00014      *               Param_L, Param, Opt_size
00015      *               InPipeN, OutPipeN)
00016      *   Name is any length, but has a valid terminator
00017      *   (high bit set on last byte, or delimiter after it)
00018      *
00019      *   Process_Num byte field, process number of new task.
00020      *   Lang_Type byte field, language/type byte for
00021      *   forked module.
00022      *   Param_L, integer field, length of parameter area.
00023      *   Param field of any type, parameter area to be
00024      *   passed to forked process.
00025      *   Opt_Size byte field, optional data area size in
00026      *   pages.
00027      *   InPipeN, integer field, path number
00028      *   OutPipeN, integer field, path number
00029      *   Process_Num, InPipeN, OutPipeN, and Return_Code
00030      *   are altered by StrtTask, no other parameters are.
00031      *

```

```

IFPI
ENBC

#####
# Offsets to arguments
#
ACount equ 2
ModuleW equ 4
ProcName equ 8
ModType equ 12
ParamLen equ 16
Param equ 20
ModSize equ 24
InPipeW equ 28
OutPipeW equ 32

Type set SBRTN+OBJCT
Revs set REENT+1
StdIn equ 0
StdOut equ 1

97CD00B1 ood TLen,StrtTask,Type,Revs,SEntry,0
53747274 StrtTask fcs /StrtTask/
2F504950 Pipe fcs "/PIPE"
01 fcb 1 version

SEntry

EC62 ldd ACount,S get param count
10B30008 cmpd 0B are there 8 params?
102600B3 lbne BadExit no; leave now.

# Set up Pipes for StdIn and StdOut.

```

```

00062      *   The procedure is:
00063      *       Dup the stdin and stdout paths to save them.
00064      *       Close stdin and stdout.
00065      *       Open /PIPE twice. One will be path 0 the next
00066      *       path 1.
00067      *       Fork the new process.
00068      *
00069      * -----
00069      * Bitsets from S for local storage
00070      0000      DStdIn  equ  0
00071      0001      DStdOut equ  1
00072      0002      LocalSiz equ  2
00073      *
00074      0025 327E      leas -LocalSiz,S make space for temp storage
00075      0027 8600      lda  #StdIn
00076      0029 103F82     D$9  l$dup      Dup StdIn
00077      002C 2570      bcs  BadExit2
00078      002E A7E4      sta  DStdIn,S
00079      0030 8601      lda  #StdOut
00080      0032 103F82     D$9  l$dup      Dup StdOut
00081      0035 2574      bcs  BadExit2
00082      0037 A761      sta  DStdOut,S
00083

```



```

00084 0039 B600      lda 0StdIn
00085 0038 103FBF     059 10Close   Close StdIn
00086 003E 2568      bcs BadExit2
00087 0040 B601      lda 0StdOut
00088 0042 103FBF     059 10Close   Close StdOut
00089 0045 2564      bcs BadExit2
00090
00091 0047 30B0FFCA    leax Pipe,PCB
00092 0048 B603      lda 0UPDAT.
00093 0040 103FB4     059 10Open    Open a pipe in path 0
00094 0050 2559      bcs BadExit2
00095      * This will be path 0
00096
00097 0052 30B0FFBF    leax Pipe,PCB
00098 0056 B603      lda 0UPDAT.
00099 0058 103FB4     059 10Open    Open a pipe in path 1
00100 005B 254E      bcs BadExit2
00101      * This will be path 1
00102 005D 103FB2     059 10Dup     Dup it
00103 0060 2549      bcs BadExit2
00104 0062 A7FB22     sta [LocalSize+OutPipeN,S]
00105
00106 0065 B600      lda 0StdIn
00107 0067 103FB2     059 10Dup     Dup it
00108 006A 253F      bcs BadExit2
00109 006C A7FB1E     sta [LocalSize+InPipeN,S]
00110
00111 006F AE66      ldx LocalSize+ModuleN,S address of module name
00112 0071 10AEFB12    ldy [LocalSize+ParamLen,S] length of parameters
00113 0075 A6F80E     lda [LocalSize+ModType,S] type of module to invoke
00114 0078 E6FB1A     ldb [LocalSize+ModSize,S] optional data area size
00115 007B ECEB16     ldu LocalSize+Param,S pointer to parameters
00116 007E 103F03     059 F0Fork   start the new process
00117 0081 2528      bcs BadExit2
00118 0083 A7FB0A     sla [LocalSize+ProcNum,S] save new process number
00119
00120      * Restore the original stdin and stdout files to
00121      * paths 0 and 1.
00122
00123 0086 B600      lda 0StdIn   Close StdIn and StdOut
00124 0088 103FBF     059 10Close
00125 008B B601      lda 0StdOut
00126 008D 103FBF     059 10Close
00127 0090 A6E4      lda 0StdIn,S   path number of duped stdin
00128 0092 103FB2     059 10Dup     dup it into path 0
00129 0095 A6E4      lda 0StdIn,S
00130 0097 103FBF     059 10Close   and close it
00131 009A A661      lda 0StdOut,S   path number of duped stdout
00132 009C 103FB2     059 10Dup     dup it into path 1
00133 009F A661      lda 0StdOut,S
00134 00A1 103FBF     059 10Close   and close it
00135 00A4 3262      leas LocalSize, clear stack
00136 00A6 5F        clrb     clear carry
00137 00A7 39        rts      return
00138 00A8      BadExit
00139 00AB 43        coma     set carry
00140 00A9 327E      leas -LocalSize,S dummy push
00141 00AB      BadExit2
00142 00AB 3262      leas LocalSize, clear stack
00143 00AD 39        rts      return
00144 00AE 239951     EMOO
00145 00B1      TLen     equ 0
00146      tll      Wait for a (child) process to complete
00147      nan      WaitTask
00148
00149      * WaitTask is a subroutine for Basic09
00150      * Wait for the a child process to complete.
00151      * Return the process ID of the process that completed
00152      * in parameter one.
00153      * Return the completion code of the process
00154      * in parameter two.
00155      * This subroutine will wait using no CPU time until
00156      * a child process completes.
00157      * If a child completed just before WaitTask was
00158      * called, it will return almost immediately.
00159      * If there are no children, an error will be returned
00160      * with a process number of 0.
00161      * Calling sequence:
00162      * RUN WaitTask (Process_No, Comp_Code)
00163      * both process_no and Comp_Code are BYTE variables.
00164

```

```

00165 0021      Type set SBATH=OBJECT
00166 0081      Revs set REENT=)
00167 0000 07C00032    mod MLen,WaitTask,Type,Revs,WaitEntry,0
00168 0000 57616974    WaitTask fcs /WaitTask/
00169 0015 01          fcb 1      addition
00170 0016      WEntry
00171 0016 6FF804      clr 14,S1   zero the process ID
00172 0019 EC62      ldd 2,S      param count
00173 0018 10B30002    cmpd 42     if not exactly 2 params then
00174 001F 260C      bne WBEit2   the caller is making a bad mistake
00175 0021 103F04     059 F0Wait   wait for a child
00176 0024 2508      bcs WBEit
00177 0026 A7FB04     sla 14,S1   return the process ID
00178 0029 E7FB08     stb 10,S1   return the completion code
00179 002C 39        rts      return
00180 0020      WBEit2
00181 002B 43        coma     set carry
00182 002E      WBEitL
00183 002E 39        rts      return
00184 002F 4C34C4     EMOO
00185 0032      MLen     equ 0
00186      end

```

```

00000 error(s)
00000 warning(s)
000E3 00227 program bytes generated
00000 00000 data bytes allocated
02188 08587 bytes used for symbols

```

```

1 #include <stdio.h>
2 #define VDIMENSION 24
3 #define MDIMENSION 80
4 #define BYTES MDIMENSION/8
5 #define TRUE 1
6 #define FALSE 0
7
8 /*----- Data Structure -----*/
9 /* The rasterized data is kept in an array of bits.
10 /* The Setbit and BitSet routines are responsible for
11 /* determining which bit corresponds to each
12 /* position. They also are the only procedures with
13 /* access to the "bit" array.
14 /*-----*/
15 main()
16 {
17 int x1, y1, x2, y2;
18 int i;
19 char opt; /* takes values of L Line (n,n,n,n)
20           C Circle (n,n,0,0)
21           S Spline (open) (n,n,n,n,n,n)
22           E Spline (closed) (n,n,n,n,n,n,n)
23
24 register int j;
25
26 while (scanf("%c %d %d %d %d", &opt, &x1, &y1, &x2, &y2) != EOF)
27 /* ignore "op" for now */
28 if (x1 < x2)
29 draw(x1, x2, y1, y2);
30 else
31 draw(x2, x1, y2, y1);
32
33 for (i=VDIMENSION-1; i>0; i--)
34 {
35 for (j=0; j<MDIMENSION; j++)
36 putchar(bitset(j,i) ? '1' : '0');
37 printf("\n");
38 }
39 return;
40 } /* end of main */
41
42 draw(x1, x2, y1, y2)
43 int x1, x2, y1, y2;
44 {
45 int dxdy, dydx, x, y, dx, dy;
46 float e, slope;
47 register int i;
48

```

```

49  delay = y2-y1;
50  deltas = x2-x1;
51  x = x1;
52  y = y1;
53  if (deltas == delay) || (deltas == 0)
54  { /* special case -- draw a point */
55      plot(x,y);
56      return;
57  }
58
59  if (deltas > delay)
60  {
61      if (deltas == 0)
62          /* prevent division by zero */
63          y = (y1 <= y2) ? y1 : y2;
64          for (i=0; i<=(deltas >= 0) ? deltas : -deltas; i++)
65              plot(x,y++);
66          return;
67      }
68      slope = (float)deltas/(float)deltas;
69      if (slope >= 0)
70      {
71          e = slope-0.5;
72          dy = 1;
73      }
74      else
75      {
76          e = slope+0.5;
77          dy = -1;
78      }
79      for (i=0; i<=deltas; i++)
80      { /* actually draw the line */
81          plot(x,y);
82          if ((slope > 0.0) || (e>0.0)) {
83              if (slope < 0.0) || (e<0.0)) {
84                  y += dy;
85                  e -= dy;
86              }
87              x++;
88              e += slope;
89              /* actually draw the line */
90          }
91      }
92      else
93      {
94          slope = (float)deltas/(float)deltas;
95          if (slope > 0)
96          {
97              e = slope-0.5;
98              dx = 1;
99          }
100          else
101          {
102              e = slope+0.5;
103              dx = -1;
104          }
105          for (i=0; i<=deltas; i++)
106          { /* -----
107              * draw a line with slope greater than one *
108              * for this type of line y needs to be *
109              * incremented more frequently than x. *
110              * ----- */
111              plot(x,y);
112              if ((slope > 0) || (e>0)) || ((slope < 0) || (e<0))
113              {
114                  x += dx;
115                  e -= dx;
116              }
117              y++;
118              e += slope;
119          }
120      }
121      return;
122  } /* end of draw */
123
124  plot(x,y)
125  int x,y;
126  {
127      setbit(x,y);
128      return;

```

```

129  }
130
131  static char bit[VDIMENSION][BYTES];
132
133  setbit(x,y)
134  int x,y;
135  {
136      int temp=1;
137      register int tx;
138
139      temp = temp << (x/8);
140      tx = x/8;
141      bit[y][tx] = bit[y][tx] | temp;
142      return;
143  }
144
145  hitset(x,y)
146  int x,y;
147  {
148      int temp=1;
149
150      temp = temp << (x/8);
151      return (bit[y][x/8] & temp);
152  }
153

```

A BASIC9 FINANCIAL MODELING PROGRAM

Ray Bovey
772 Gapter Road
Boulder, CO 80303

COMPSHEET

Have you ever had trouble convincing people that your home computer is really useful? I am absolutely certain that they are great, but I sometimes have a little bit of difficulty convincing my wife of their value. Of course it doesn't help that she dislikes numbers, hates math, and feels that it is terrible to spend perfectly nice days (of which we have quite a few here in Colorado) cooped up indoors typing at a terminal. Besides, I have to admit, not all of the things I do with my machine are strictly useful. Take the voice synthesizer for example. It was fun to type in a half page of funny characters and then hear it try to sing Happy Birthday, but it didn't really serve any very useful purpose.

Given this background you will no doubt appreciate that I am always eager to find some useful task for my home computer to perform. And, amazingly, the chance finally came about a year ago. My wife signed up for a class on Financing Real Estate only to discover that what she had thought would be a fun and interesting course was full of numbers and calculations (the professor casually suggested that all the students buy \$150 HP Financial calculators). One Saturday, after I had spent all afternoon (on a perfectly nice Colorado day, mind you) helping her go through detailed calculations showing how to make the most profit on financing a shopping center, it occurred to me that this was the sort of thing that computers were invented for. All we needed was one of those nifty spreadsheet programs. But how could I convince my wife that it made sense to spend considerably more than \$150 on a spreadsheet program that she couldn't even take to class with her? "That's easy! I'll just write one myself. You can't trust other people's programs anyway", I said to myself.

I promised it wouldn't take more than about 2 hours (she was skeptical. She had to turn in the results Monday night and Saturday was already gone). Well, as you can certainly imagine, these things don't always go as smoothly as one might expect. I worked all day Sunday and late into the night. Finally I helped her do the rest of the calculations on a calculator. But I was determined to finish up the spreadsheet program. Besides, she still had a term project due at the end of the semester. The final result is the COMPSHEET program presented here. In the rest of this article I will give you an overall description of the program and an illustrated example of its use.

Features of COMPSHEET

The first feature of COMPSHEET is that it is cheap (you can type in the accompanying listing or for \$10 I will send you a 5" soft sector 05-9 floppy with the program

on it). It is definitely not a full-fledged spreadsheet program but it does help quite a bit in doing complicated financial calculations and projections over a period of years. On my 56K OS-9 (trademark of Microware) Level I machine, the program can only handle 30 rows of 20 columns each, but that is plenty for many situations. If you have a Level II system, you could no doubt increase this.

The program is written in Microware's excellent BASIC09 (trademark of Microware and Motorola) language. A few of the features of this BASIC which are used in the program include user defined variable types (which can be used like PASCAL records) and a true subroutine capability (none of this hokey GOSUB stuff for me!). The ample collection of "structured programming" constructs available in BASIC09 is also put to good use. You will note the absence of any program numbers in the program. Another substantial advantage of BASIC09 is its impressive speed. There is very little delay while COMPSHEET does its calculations. Obviously, writing a program in BASIC makes it much easier to try out changes and improvements to the code.

The program allows you to give a name (which can be used in formulas to calculate the values of future rows), a title (which will be printed out), a method of calculation for the values to be placed in the row (more about this in a minute), and miscellaneous formatting instructions pertaining to how the row will appear on the output. The values for a given row can be specified directly with a list of numbers, or you can specify the number for the first column in the row and let COMPSHEET add an increment or inflation factor to this to calculate values for the remaining columns in the row. The final choice is to give COMPSHEET a formula involving previous rows and or numbers. Formulas can include the four basic arithmetic operators +, -, *, and /. You can also specify title lines or other alphanumeric information to make the final spreadsheet clear to the user (remember, my wife had to understand the results of all this).

Three formats are provided for output numbers - dollar (commas are inserted as appropriate and negative numbers are enclosed in parentheses), numeric (just plain old numbers), and percent (this also affects how input numbers are treated). Of course, columns are nicely justified to make things look neat.

One thing the program doesn't do is to allow you to move around on the screen and change things interactively (I told you this was not a full-blown commercial spreadsheet). Once you have typed something in you are stuck with it. However, all of your input is automatically saved in a file so that you can edit it later. "Wouldn't it be nice if I could afford an editor?", you say. With COMPSHEET that is no problem! The file that your input is saved in can be edited with the editor that comes as a part of BASIC09. Once you have made any changes to your input you can run COMPSHEET on the file. As it reads through the file the program lists the lines to your terminal so that you can see what it is working on.

A Practical Example

In order to keep things simple I won't go through a whole shopping center example. Let's just suppose that you are thinking of buying some vacant land with the intention of subdividing it into building lots in a couple of years. The land will cost you \$50,000 and you can get a 25 year 14% loan on it for 80% of that amount (don't ask me how or where). Further, we will assume that with about \$10,000 of development expenses you could resell the land as three lots for \$30,000 each in the third year of ownership. There are several numbers we need to calculate before we are ready to fire up COMPSHEET. First, we need to know what yearly payments on the loan will come to, and how much of these will represent payment of interest (interest payments are tax deductible). This can be found from a standard amortization table or a simple program. In our case, yearly payments (referred to as debt service in real estate jargon) come to \$5778. Of this amount, \$5588 goes to interest in the first year, \$5560 goes to interest in the second year, and \$5527 in the third year. The difference between these amounts and the yearly debt service payments represents paydown of the principal and amounts to a grand total of \$659 over the first three years.

Now, with these numbers in hand, we are ready to begin our calculations. First, we will calculate before tax cash flows, then the Federal Income Tax ramifications of the investment, and finally the all-important after tax cash flow (ATCF).

From a cash point of view the initial investment is the difference between the total cost of the land (\$50,000) and the amount of your loan (\$40,000). Although this number is not used in calculating cash flows it is certainly useful to bear in mind.

For the first two years the cash flow will be comprised only of the annual debt service (in most areas of the country you would also have to pay property taxes, but we will leave those out for simplicity). However, in the third year, there will be some income (estimated at \$90,000) from the sale of the lots. In addition, we anticipate expenses of \$10,000 to develop the lots. Combining all of these items we calculate the before tax cash flow which, for reasons which are unclear, is generally termed the cash throwoff to equity or CTOE.

Now we must deal with the tax consequences of this investment. These consist of a deduction for our interest expenses and longterm capital gains treatment for our profit (which is charged at only 40% of the rate for normal income). The profit consists of the \$90,000 we sell the lots for minus the original price of \$50,000 we paid minus our \$10,000 development expenses (note that only the interest portion of our debt service payments counts as an expense - the rest went into paying off the mortgage). So we end up with a \$30,000 longterm capital gain (LTCG). Combining the interest deductions and the LTCG we calculate the annual taxable amount relative to this investment and assume we are in the 50% tax bracket in order to figure the final tax liability (or savings). This amount is then combined with the CTOE calculated previously to yield the ATCF.

Let me make it clear that I am not an accountant and the details of the above tax treatment may not be entirely accurate. However they serve adequately as an example of what COMPSHEET can do for you.

Listing of Input to COMPSHEET

PROCEDURE example

```
0000 REM ^3
0005 REM ^15
0008 REM ^58
0011 REM ^center
0018 REM ^COMPSHEET Example
0030 REM ^center
003A REM ^=====
004F REM ^literal
005A REM ^
005E REM ^literal
0069 REM ^Assumptions :
007A REM ^literal
0085 REM ^
0089 REM ^literal
0094 REM ^ Initial purchase price of $50,000.
00C8 REM ^literal
00CB REM ^ Loan for $40,000 at 14% over 25 years.
00FB REM ^literal
0106 REM ^
010A REM ^literal
0115 REM ^
011A REM ^literal
0125 REM ^ Year 1 Year 2 Year 3
0158 REM ^underline
0165 REM ^row
016C REM ^income
0176 REM ^Sale of lots
0186 REM ^0
018B REM ^0
0198 REM ^90000
0199 REM ^row constant
01A9 REM ^ds
01AF REM ^- Debt service
01C1 REM ^-5778
01CA REM ^row
01D1 REM ^devel
01DA REM ^- Development expenses
01F4 REM ^0
```



```

01F9 REM ^0
01FE REM ^-10000
0200 REM ^row
020F REM ^repay
0210 REM ^- Loan repayment
022C REM ^0
0231 REM ^0
0236 REM ^-39341
0240 REM ^underline
024D REM ^row formula
025C REM ^ctoe
0264 REM ^CTOE
026C REM ^income + ds + devel + repay
0200 REM ^literal
0296 REM ^
0290 REM ^row hide
02A7 REM ^gain
02AF REM ^
0203 REM ^0
0200 REM ^0
020D REM ^90000 - 50000 - 10000
02D6 REM ^row formula
02E5 REM ^ltg
02EC REM ^Longterm gain @ 40%
0303 REM ^gain * 0.40
0312 REM ^row
0319 REM ^int
0320 REM ^- Interest
032E REM ^-5500
0337 REM ^-5560
0340 REM ^-5527
0349 REM ^underline
0356 REM ^row formula
0365 REM ^taxamt
0370 REM ^Taxable amount
0302 REM ^ltg + int
030F REM ^row constant percent
03A7 REM ^taxrate
0302 REM ^Tax rate
030E REM ^50
03C4 REM ^underline
03D1 REM ^row formula
03E0 REM ^taxliab
03EB REM ^Tax liability
03FC REM ^taxamt + taxrate
0411 REM ^literal
041C REM ^
0421 REM ^row formula
0430 REM ^
0435 REM ^CTOE
043D REM ^ctoe
0445 REM ^row formula
0454 REM ^
0459 REM ^- Tax liability
046C REM ^-taxliab
0470 REM ^underline
0405 REM ^row formula
0494 REM ^atcf
049C REM ^ATCF
04A4 REM ^ctoe - taxliab

```

Listing of Output from Example

COMPSHEET Example

=====

Assumptions :

Initial purchase price of \$50,000.
Loan for \$40,000 at 14% over 25 years.

	Year 1	Year 2	Year 3

Sale of lots	0	0	90,000
- Debt service	(5,770)	(5,770)	(5,770)
- Development expenses	0	0	(10,000)
- Loan repayment	0	0	(39,341)

CTOE	(5,770)	(5,770)	34,001
Longterm gain @ 40%	0	0	12,000
- Interest	(5,500)	(5,560)	(5,527)

Taxable amount	(5,500)	(5,560)	6,473
Tax rate	50.00%	50.00%	50.00%

Tax liability	(2,794)	(2,700)	3,237
CTOE	(5,770)	(5,770)	34,001
- Tax liability	2,794	2,700	(3,237)

ATCF	(2,904)	(2,990)	31,645

Comments on Example

Refer to the listing included of Input to COMPSHEET in this discussion of the example. The listing was produced inside BASIC09 as the memory locations referenced provide an easy way to refer to individual lines within the file. Note that each line begins with REM 0. This is added by COMPSHEET at the beginning of each line that you type in to make it look like a valid BASIC09 statement so that you can use the editor inside BASIC09 to edit the file. The first three lines describe the number of columns, how many characters to allow for alphanumeric labels on each row, and the total paper width to use. After this, succeeding lines alternate between COMPSHEET commands (such as CENTER, LITERAL, ROW, or NUMBER) and the data needed by these commands. In the case of the UNDERLINE command, no data is needed. For CENTER and LITERAL one line of text

constitutes the data, while for ROW or NUMBER the data consists of a line with the item name (which if nonblank can be used later to refer to this item in formulas), then a line with the label to print at the left of this item, and finally the numbers for this item (or the formula or a base value and increment or a base value and percentage increase depending on which option was selected along with the command).

Of special interest are the lines at 02 B - 02BD. Here we declare a row as "HIDE". This tells COMPSHEET to go ahead and calculate everything for this row, but not to display it. This feature can be used to simplify the visual appearance of the output. Note also that the labels specified at lines 01DA and 02EC are neatly split at word boundaries since they are too long to fit within the number of characters we specified for labels.

SUMMARY

This program does not claim to provide the features of a commercial spreadsheet program. However, I believe that many readers may find it useful for relatively simple computations. It provides a neat output with several different options for number formatting. Features such as the powerful editor built into BASIC09 help to keep the program simple, and enables it to fit entirely in memory. Because the program takes advantage of BASIC09's ability to handle separate subprograms it will be relatively easy for users to customize the package to their own needs. Anyone interested in getting a 5 inch floppy containing this program may do so by sending \$10 to the author.

Listing of COMPSHEET

```

PROCEDURE compsheet
TYPE stoa,typenumber;STRING(10); values(20);REAL
DIM lline$;STRING,tyoa
DEF The above 2 lines allow up to 20 years and 30 llines
DIM command$;STRING(10)
DIM qualifiers(8);INTEGER
DIM qt,q2,totl;INTEGER
DIM lline,outline,compline;STRING(255)
DIM fillo,out,prti;INTEGER
DIM fillo;BOOLEAN
DIM title_size,column_size;INTEGER
DIM nyr,nl;INTEGER
nyr=1
DIM i,j,b,lines;INTEGER
DEF Find out whether they want input from terminal or file,
PRINT "Do you have an existing file for input ?"
INPUT c$
IF LEFT$(c$,1)="" ON LEFT$(c$,1) THEN
PRINT "Filename ?"
INPUT c$
OPEN fillo,c$;READ
fillo=TRUE
READ fillo,linel;WHEN slip over dummy Procedure line
CREATE out,"term";WRITE
ELSE
PRINT "Filename to save in ?"
INPUT c$
OPEN fillo,"term";UPDATE
fillo=FALSE
CREATE out,c$;WRITE
WRITE out,"PROCEDURE "c$;WHEN Write out dummy Procedure line
ENDIF
RUN getintofillo,fillo,out,nyr,"How many years (1-20) ?"
RUN getintofillo,fillo,out,tlsize,"How many columns for names ?"
)
column_size=column_size-tlsize;slip nyr
DATA "LITERAL","CENTER","UNDERLINE","NUMBER","ROW"
FOR i=1 TO 5
READ command$
NEXT i
DATA "LIST","CONSTANT","INCREMENT","INFLATE","FORMULA","DOLLARS"
,"PERCENT","NUMERIC"
FOR i=1 TO 8
READ qualifiers(i)
NEXT i
)lines=1
CREATE prti,c$;prti;WRITE WHEN Create file to be printed later
WHILE NOT(EOF(fillo)) DO WHEN Here is the main loop of processing command lines
RUN getintofillo,fillo,out,linel,TRUE,"Command : " WHEN Get next command
)BURSTW: " ,linel;WHEN Find end of first word
IF j<0 THEN j=LEN(linel);WHEN There say only be 1 word
ENDIF
code=LEFT$(linel,j-1);WHEN Extract command word
lino=MOD(linel,j-1,265)-1

```

```

FOR j=1 TO 5 WHEN Check against legal commands
EXITIF code=command$(j) THEN j=)
ENDIF
IF j=5 THEN j=0 WHEN But 1 = 0 to indicate no action
ENDIF
NEXT j
IF j=1 THEN WHEN LITERAL command
RUN getintofillo,fillo,out,linel,FALSE,"Literal line : "
PRINT prti,linel;WHEN Print it out
ENDIF
IF j=2 THEN WHEN CENTER command
RUN getintofillo,fillo,out,linel,FALSE,"Line to center : "
WHEN Calculate how many leading blanks we need
j=(title_size+prti_size-LEN(linel))/2+1
PRINT prti,totl;j;linel
ENDIF
IF j=3 THEN WHEN UNDERLINE command
FOR j=1 TO title_size+prti_size
PRINT prti," "
NEXT j
PRINT prti,""
ENDIF
IF j=4 OR j=5 THEN WHEN NUMBER or ROW command
q1=WHEN 1 => calculation of value by LIST of values
q2=WHEN 1 => format is DOLLARS by default
q3=WHEN 1 => HIDE is false by default
FOR i=1 TO 5 WHEN Compare for calculation of value qualifier
)BURSTW(qualifiers(i),linel)
IF k>0 THEN
q1=)
ENDIF
NEXT i
FOR j=4 TO 5 WHEN Compare for format qualifier
)BURSTW(qualifiers(i),linel)
IF k>0 THEN
q2=)
ENDIF
NEXT j
IF BURSTW("HIDE",linel) THEN q3=0 WHEN Check for HIDE qualifier
ENDIF
RUN getintofillo,fillo,out,linel,TRUE,"Line name : "
lino=MOD(linel,265)-1;WHEN Keep only first 10 chars
RUN getintofillo,fillo,out,linel,FALSE,"Line title : "
IF c3=0 THEN WHEN If not hiding list out title
PRINT prti," "
WHEN If title is too long for 1 line break it at word boundary
WHILE (LEN(linel))>title_size DO
j=title_size+1;WHEN Assume first word of title is too long
FOR j=title_size TO 1 STEP -1
EXITIF MID$(linel,j,1)="" THEN j=j-1
ENDIF
NEXT j
IF j<title_size THEN WHEN We found a break point
PRINT prti,LEFT$(linel,j-1)
lino=MOD(linel,j-1,265)-1
ELSE WHEN We lost hyphenation. Don't try to use rules
PRINT prti,LEFT$(linel,j-2)+""
lino=MOD(linel,j-2,265)-1
ENDIF
ENDIF
ENDWHILE
PRINT prti,totl;linel;tab(title_size+1);WHEN Finish title & space over
ENDIF
IF q1<5 THEN WHEN If LIST of values, CONSTANT, INCREMENT, or INFLATE
RUN getintofillo,fillo,out,linel,TRUE,"Initial value : "
RUN evaluate(lino,l,lines,(lino,base_val)
IF q2=2 THEN base_val=base_val*100;WHEN Divide by 100 for percent
ENDIF
ENDIF
IF j=4 THEN WHEN NUMBER line
IF q1=5 THEN WHEN FORMULA
RUN getintofillo,fillo,out,linel,TRUE,"Give the formula : "
RUN evaluate(lino,l,lines,(lino,base_val)
ENDIF
FOR j=1 TO nyr WHEN For NUMBER each year in the name
lino=lino+1;slip;j=base_val
NEXT j
IF q3=0 THEN WHEN If not HIDE
RUN formatprti,q2,l,column_size,lino;linel)
PRINT prti,"" WHEN End the line
ENDIF
WHEN Only remember this lino if they gave it a name
IF lino=linel.name THEN
)name=linel.name
ENDIF
ENDIF
IF j=5 THEN WHEN ROW
IF q1=1 THEN WHEN LIST of Values
lino=lino+1;slip;j=base_val
FOR j=2 TO nyr
PRINT USING "Value for year",j,q3,j;" " ;
RUN getintofillo,fillo,out,linel,TRUE,""
RUN evaluate(lino,j,lines,(lino,base_val)
NEXT j
WHEN Divide by 100 if this is in PERCENT

```

```

IF a2=2 THEN item(lineno).value(j)=item(lineno).value(j)+1
ENDIF
NEXT j
ENDIF
IF a1=2 THEN OPEN CONSTANT
FOR j=1 TO nym
item(lineno).value(j)=base_val
NEXT j
ENDIF
IF a1=3 THEN WHEN INCREMENT
RUN getlinchfilo,filo,out,inline,TRUE,"Increment s "
RUN evalstartinline,i,lineno,itoo,incr
incr=0
FOR j=1 TO nym
item(lineno).value(j)=base_val+incr
incr=incr+loc
NEXT j
ENDIF
IF a1=4 THEN WHEN INFLATE
RUN getlinchfilo,filo,out,inline,TRUE,"Inflate rate t "
RUN evalstartinline,i,lineno,itoo,incr
incr=0
incr=incr*filo.
FOR j=1 TO nym
item(lineno).value(j)=base_val+incr
incr=incr+inc
NEXT j
ENDIF
IF a1=5 THEN WHEN FORMULA
RUN getlinchfilo,filo,out,inline,TRUE,"Formula i "
temploos=inline WHEN Save the formula
FOR j=1 TO nym
inline=temploos
RUN evalstartinline,j,lineno,itoo,base_val
item(lineno).value(j)=base_val
NEXT j
ENDIF
IF a3=0 THEN WHEN Not HIDE
RUN forselcort,02,nym,column_size,item(lineno)
PRINT sp,t," WHEN End the line
ENDIF
WHEN Only remember this line if they gave it a name
IF item(lineno).name="" THEN
lineno=lineno+1
ENDIF
ENDIF
IF a=0 THEN WHEN Here then end of line
PRINT "Input line contains no command; ignored i " i inline
ENDIF
ENDIF
ENDWHILE
CLOSE filo
CLOSE fout
CLOSE fout
DIE
PROCEDURE getlin
PARAM filo:WORD
PARAM filo,out:INTEGER
PARAM inline:STRING(263)
PARAM nycas:BOOLEAN
PARAM prompt:STRING
DIM i,j:INTEGER
inline=""
IF EOF(filo) THEN END
ENDIF
IF filo THEN
READ filo,inline
inline=MID(inline,0,260-65) WHEN Strip off dummy REN "
WRITE fout,inline
ELSE
PRINT filo,prompt
READ filo,inline
WRITE fout,"REN "+inline WHEN Insert dummy REN "
ENDIF
IF nycas THEN
RUN CUDUP(filo)
ENDIF
END
PROCEDURE token
PARAM inline:STRING(263)
PARAM totype:INTEGER
PARAM str:STRING
PARAM value:REAL
DIM i,j:INTEGER
i=1
str=""
value=0
IF inline="" THEN
totype=0 WHEN no token. Just return
END
ENDIF
WHEN Strip leading blanks
WHILE LEFT(inline,i)="" DO
inline=MID(inline,2,264)
ENDWHILE

```

```

WHILE LEFT(inline,i)
WHEN Check for item name
IF "A"<=c AND c<="Z" THEN
tottype=2 WHEN item name found
str=""
WHEN Retrieve rest of item name
WHILE "A"<=c AND c<="Z" DO
str=str+c:next=c
inline=MID(inline,2,264)
c=LEFT(inline,i)
ENDWHILE
ELSE
WHEN Other possibilities are a number or an operator
IF "0"<=c AND c<="9" OR c="." THEN
totype=1 WHEN a number
value=0
WHEN Handle leading digits
WHILE "0"<=c AND c<="9" DO
value=value*10+ASC(c)-ASC("0")
inline=MID(inline,2,264)
c=LEFT(inline,i)
ENDWHILE
WHEN Check for decimal point.
IF c="." THEN
WHEN If found, strip it out
inline=MID(inline,2,264)
c=LEFT(inline,i)
place=.1
WHEN and handle fraction digits.
WHILE "0"<=c AND c<="9" DO
place=place*.1
value=value+(ASC(c)-ASC("0"))*place
inline=MID(inline,2,264)
c=LEFT(inline,i)
ENDWHILE
ENDIF
ELSE
totype=3 WHEN a operator
str=c
inline=MID(inline,2,264)
c=LEFT(inline,i)
ENDIF
END
PROCEDURE copper
PARAM inline:STRING(263)
DIM i,j:INTEGER
n=LEN(inline)
FOR i=1 TO n
c=MID(inline,i,1)
IF "A"<=c AND c<="Z" THEN
inline=LEFT(inline,i-1)+CHR(ASC(c)+ASC("A")-ASC("a"))+MID(inline,i+1,n-i+1)
ENDIF
NEXT i
END
PROCEDURE evaluate
TYPE item_type=INTEGER value:REAL
PARAM inline:STRING(263)
PARAM str:INTEGER
PARAM lineno:INTEGER
PARAM itoo:INTEGER_type
PARAM value:REAL
DIM i,totype:INTEGER
value=0 WHEN Start off as 0. We will add to it.
opt="" WHEN implied addition before first operand
RUN token(inline,totype,c,value)
WHILE totype>0 DO WHEN While there are tokens on the line
IF totype=3 THEN WHEN OPERATOR
opt=c WHEN Save operator for later use.
ELSE
IF totype=2 THEN WHEN item NAME
FOR j=1 TO lineno-1 WHEN Find this name in our list.
EXIT IF item(j).name=c THEN value=item(j).value:break
ENDIF
IF j=lineno-1 THEN
PRINT "Formula involved unrecognized item name i " i c
value=0
ENDIF
NEXT j
ENDIF
WHEN If it was either a NAME or a NUMBER we can now apply the
WHEN saved operator to it.
IF opt="" THEN
value=value+value
ELSE
IF opt="*" THEN
value=value*value
ELSE
IF opt="/" THEN
value=value/value
ELSE
IF opt="+" THEN
value=value+value
ELSE
IF opt="-" THEN
value=value-value

```


EPSON HX-20 REVIEW

Peter A. Stark
Star-Kits
P. O. Box 209
Mt. Kisco, NY 10549

Why review the Epson HX-20 computer in 68 Micro, you ask? Simple - the HX-20 uses the 6301 microprocessor. The 6301 is a CMOS version of the 6803 processor, which is quite similar to the 6800 and definitely a member of the 68xx family.

Like the 6803, the 6301 can run 6800 machine language programs, but it also contains several additional instructions which the 6800 does not have. These are more like those of the 6809. For example, it has ABX (add B to X); LDD, ADDD, STD and other double-accumulator instructions; and PSHX and PULX instructions for saving the contents of the index register on the stack. In addition, the 6301 has a few instructions which even the 6803 does not have - a "sleep" instruction and several bit manipulation instructions. In addition, of course, it is CMOS and so it can easily be used in battery operated systems. The HX-20 is what some call a "portable notebook" or "briefcase" computer, a bit smaller than a 2" thick loose-leaf binder. It has an internal nickel cadmium battery, and thus the 6301 is an ideal processor for such a computer.

The HX-20 sports two of these processors - one for general computing, and the other for controlling some of the specialized I/O devices. In addition, it comes with 16K bytes of RAM (expandable to 32K with a \$150 plug-in memory expansion module), and an internal time of day clock. It also includes 40K of ROM, part of which is bank-switched. It comes with a Microsoft Basic Interpreter and also an 8K word processor called SkiWriter.

I/O Capability

The HX-20 has a standard typewriter-size keyboard with 56 keys, four additional cursor control keys, three program control buttons, and five function buttons. The keyboard is quite complete, and even includes square and curly brackets, the inverse slash, and other special characters. It can also generate control characters with its CTRL key.

The display is an LCD display of 4 lines by 20 characters across. An angle adjustment control makes the display easy to read even when viewed at strange angles. Though this is a fairly limited display, the HX-20 maintains a "virtual screen" which is typically eight lines by 40 characters, and whose size can be varied. The actual LCD screen can be thought of as a window on this virtual screen, and the cursor control keys can move the window back and forth over the virtual screen. Though this does not solve all the problems of a small display, still it does have many advantages. It's quite handy to be able to go back a few lines above the visible screen to look at something after it has scrolled off the LCD display.

What really makes the HX-20 unique and different from other briefcase style computers (such as the Radio Shack Model 100, which is in about the same category) is its printer and cassette data recorder. Whereas other small machines require external printers and recorders, the HX-20 has them built in.

The printer is a miniature dot matrix unit which prints on 2-1/4" wide adding machine paper. It occupies the top left corner of the computer's front panel. The printer can be used for normal printing; in addition, the Microsoft Basic has a COPY command which can be used to dump the contents

of the screen, graphics as well as text, to the printer. Though it is a bit noisy in a quiet room, the printer is not objectionably so, even when running for a while at its maximum speed, which is a bit over one line of text per second.

The second unique I/O device is a microcassette recorder which occupies the top right corner of the front panel. The recorder used to be an extra-cost option, but is now included with the HX-20 as part of the \$799 purchase price.

If you think the printer is small, the recorder is even smaller. It has no mechanical controls, since its operation is entirely under computer control. Instructions in the program can stop and start it, and the function keys also double for recorder control. Though smaller, this recorder is much more capable than the larger cassette recorders connected to other computers, since it can be fully controlled by the program.

Rather than having a mechanical counter, this recorder's counter is simply a switch which generates an electrical pulse as the tape reel turns. These pulses are kept track of by the 6301 which does I/O control. The tape count can be displayed at any time by the TAPCNT command in Basic.

The advantage of this arrangement is that the program always knows where the tape is positioned. Basic has a WIND command which can be used to either completely rewind the tape, or else to wind the tape to any specific spot on the tape. In this way it is possible to accurately place program or data files on the tape under fully automatic control.

Finally, the HX-20 has several connectors around its sides. There is a bus connector on the left side for the addition of major I/O devices. On the back there are two DIN connectors, one for RS-232C devices such as a modem or printer, the other a high-speed serial connector for a floppy disk controller or CRT interface (both of which have been promised, but not yet released). On the right side are three small connectors for an external cassette recorder and a bar code reader.

It is interesting to compare the I/O devices with those of the Radio Shack Model 100, a newer and probably much more popular machine. The Model 100 also has a full keyboard and LCD display, but its display has 8 lines instead of 4, and 40 characters per line rather than 20. The Model 100 characters also appear larger, and thus not as clearly defined.

In many applications, especially text processing, the larger LCD display of the Model 100 is noticeably better. In fact, the small display of the Epson is a serious drawback. On the other hand, I've done some text editing on the one-line display of the Radio Shack pocket computer, so I would say that it is quite possible to do respectable work on the four-line display of the Epson as well.

By having a smaller display, the Epson also has room for both a printer and the microcassette recorder, which the Model 100 lacks. Regardless of how easy or difficult it is to edit text, it is always useful to be able to get a printed listing, and (in my mind) essential to get it stored on some mass storage device before you accidentally clobber it. While a printer and cassette recorder can be connected to the Model 100, it is obviously much more awkward to carry three boxes than to carry just one.

The Model 100, on the other hand, has a built-in direct connect modem. Several writers in other magazines have indicated that they like to do their text editing on the Model 100, and as soon as memory is full they then upload it to CompuServe or the Source, or perhaps their office computer via a phone. Except for the cost that seems like a good idea, unless you are in a place where there either is no phone, or where the phone does not have a modular jack for connection of the Model 100. That pretty much lets out pay phones and most hotel room

telephones.

The alternative, of course, is to carry on acoustic coupler. Doing that puts the Model 100 and the Epson on an almost equal footing, since you must then carry another I/O device. (I say "almost equal" because the Epson's CX-20 coupler is a more expensive option than the Model 100's coupler.) Granted that the Model 100's acoustic coupler is smaller than Epson's, I am really impressed by the Epson coupler. It is the only battery-operated coupler I know of, quite small and neat, and very well designed. It can be used in both originate and answer modes, and has a test mode as well. In fact, I am surprised that Epson has not aggressively marketed the coupler all by itself - I am sure that many owners of other computers would love to buy the Epson CX-20 coupler even though they don't have the HX-20 computer. (In fact, the CX-20 coupler may be more useful on other computers, since it is not furnished with any software for the HX-20. You would have to write your own program if you wanted to upload text files from the HX-20 to another computer.)

Software

While on the subject of comparing the HX-20 with the Radio Shack Model 100, it is interesting to compare them from the software standpoint as well.

Both computers have a Microsoft Basic Interpreter; though I have not compared them directly, I would suspect that they are both fairly similar. The Model 100, however, has several other user programs in ROM, including a telecommunications program which makes use of the built-in modem, a scheduler and address organizer, and a word processing program. The Epson, on the other hand, contains only its SkiWriter word processor program. (SkiWriter is fairly new, and earlier HX-20 computers were not equipped with it. Only recently has it been supplied as part of the computer.) The HX-20 that I used did not have the SkiWriter ROM and so I am not able to describe it; however, a reviewer writing in the September 1983 issue of MICROCOMPUTING seemed preferred the SkiWriter to the Model 100 word processor.

Whenever you turn on the computer, or whenever you press the MENU button, the LCD displays a menu with up to eight choices. The first three choices are Basic, a monitor, and (on machines so equipped) the SkiWriter editor; the remaining five choices can be Basic programs residing in RAM. You type a number from 1 to 8 to make a choice, and immediately go to the requested program.

Basic program memory can be partitioned to hold five separate programs. This memory division is dynamic, and each program gets only as much space as it needs, up to the total RAM limit available. You move from one program partition to another by a LOGIN command; assigning a TITLE to a program locks it in memory (so it cannot be accidentally erased), and puts the title in the menu.

In addition, memory can also be assigned for data storage using "RAM files". A RAM file is an area of memory that is accessed in the same way as a serial storage device which can be read or written. Unlike program variables, data in a RAM file does not get erased when you change to another program or when the computer is shut off. Hence RAM file data is more permanent, and can be shared by different programs in memory.

The Basic Interpreter itself is written by Microsoft, and has quite a few special extensions which make the HX-20 quite powerful. In addition to the standard Basic features, this Basic also has the following extra statements:

COBL, CINT and CSNG for conversions to and from Integer, real, and double precision numbers
DEFFIL, GET4 and PUT4 statements for RAM files
AUTO line numbering
COPY for dumping screen data to the printer
ERASE for erasing errors

ON ERROR GOTO, ERL and ERR recovering from errors

ERROR statement for simulating an error condition for testing

FILES for showing the contents of files

FRE for checking the amount of free memory

HEX\$ and OCT\$ functions for hexadecimal and octal numbers

Several forms of INPUT for inputting files etc.

KEY and KEYLIST for defining and reading special function keys

LINE, PSET etc. for screen graphics

LIST for listing to devices other than screen

LOF for checking the size of a file or buffer

MON for going to the monitor

RENUM for program renumbering

RESTORE for rereading DATA statements

WIND and TAPCNT for controlling the micro cassette

TITLE and LOGIN for switching program areas

TIME\$, DATE\$, and DAY for reading and setting the time of day clock

WIDTH for setting the size of the virtual screen

SOUND for controlling the piezoelectric speaker.

CONCLUSIONS

All in all, the HX-20 is a quite powerful little computer, one which shows that a lot of thought went into its design. Some of its features are quite unusual for such a small machine. For example, the keyboard has a type-ahead buffer which seems to be about 8 characters long. The piezoelectric speaker is controlled by the I/O processor, so that sound generation can be overlapped with processing. The function keys are very handy and powerful. The time of day clock can be put to a lot of good use. Above all, the presence of the built-in printer and microcassette recorder gives the computer a lot of flexibility and power it would not otherwise have.

Yet the machine has a few drawbacks which may appear serious to some. My main objection is the keyboard. Although it is full-size and looks beautiful, I find that the keys stick. I simply cannot use it at the speeds I am accustomed to. Since my main use for a computer of this type is for word processing, that rules out the HX-20 as far as I am concerned. (Considering that I have used even a Radio Shack pocket computer for text processing, my not being able to live with the Epson keyboard is not because I am picky.)

My second reservation about the HX-20 is due to the lack of software. I believe that is being currently remedied, and the availability of the SkiWriter editor is certainly an improvement. But when I first got the HX-20, the only thing you could do was to run Basic. Although I received the CX-20 acoustic coupler for testing, there was no useful software for it except for a nine-line Basic program which I accidentally found on page 390 of the manual as an example of using the LOF function.

On the other hand, the mere fact that it was on page 390 is a good sign - the documentation supplied with the computer adds up to somewhere around 700 pages, and consists of an operations manual, a Basic tutorial manual, a Basic reference manual, and a microcassette manual. Surely needed is the technical reference manual, which would hopefully provide information on the actual programming of those 6301 processors. Remember those? That's how we got interested in the HX-20 in the first place.

**SUPPORT YOUR
ADVERTISERS**

ACORN

MOTHER BOARD

REVIEW

Bare Board: \$64.95
Kit: \$149.95
Assembled: \$199.95

Why a new mother board? First, a little history. When I first got interested in computing, it was the early days. I purchased a SWTPC 6800 computer system. It came with a MP-B motherboard, a MP-A 6800 processor card (with Mikbug) and a 4k memory board with 2K supplied. The memory card was expandable by another 2K for only \$125. The whole system, at the time, cost \$395. I also purchased a CT-1024 terminal and a AC-30 cassette interface.

After much expansion and changes, the system grew to use a 6809 MP-09 processor card, dual 5" disk drives with a DC-2 controller, 64K of memory, DMAF2 controller with 8" drives and numerous peripherals such as MP-LA parallel cards for my 2 printers and the MP-T interrupt controller, and MP-R eeprom burner. I am still using a PR-40 printer which I have had for many years. It still performs well and I use it often particularly for catalog listings of my disks. The only problem here is ribbons, which have become more or less obsolete.

In this system, the mother board had been replaced by a MP-B2 SWTPC unit. Recently I added 2K of 6116 ram to the processor card for the purpose of running several small utilities as "resident". This means that the utilities are loaded into the ram by a startup file and afterwards do not have to be loaded from the disk each time you call them. They execute instantly when loaded this way. The program for this is available (public domain) from the Southern New England Flex User Group. There was, however, one problem with this approach. The ports on the MP-B2 motherboard were not fully decoded and the ports were reflected in the ram location. Out came back issues of the Micro Journal. I remembered seeing an article on fully decoding the motherboard port addresses. I found the article and modified the motherboard. This produced a kludge which looked terrible and appeared fragile as I had to solder chips onto other chips with their pins up in the air and drill the board to pass wire wrap wire underneath. Although it worked, replacing boards in the MP-B2 was a little chilling. Also my SWTPC version of FLEX would now configure for 16 bytes per port rather than the actual 4 bytes that was correct. At first I was confused and though I had made an error wiring the decoders as I could not address my printer in port seven. The SWTPC utility "SBOX" showed "10=16" and I realized the error. Using "SBOX" in a startup file cured this also.

Experiments with FLEX and other operating systems soon proved that the older motherboard had exceeded its usefulness. The limitations that I encountered in my Quest for expansion were numerous. In order to use extended addressing, the MC-14411 baud rate generator had to be removed from the processor card as the SS-50 baud rate lines were used to generate the extended addresses. The limitations of the port addressing meant that (with 2 printers and a MP-T timer) I could accommodate 5 serial ports at the most. After considering the alternatives, a newer motherboard seemed like the best idea. Unfortunately SWTPC no longer offers the chassis, motherboard and MP-ID card separately. (Even so, I estimated the cost of this conversion at over \$600.) I could get a baud rate generator for the back plane and cut the traces on the old motherboard, but this solution again reduced the number of serial ports to 4 and involved more motherboard modifications.

About this time I was introduced to Merle and Ross of Acorn Computer Systems, 11931 W. Bluemound Road in Wauwatosa, Wisconsin, a suburb of Milwaukee (Phone: 414-257-0300). I purchased a PB4 intelligent port buffer from them and replaced one MP-LA parallel port with it. The thing was very well thought out and well made. It exceeded my expectations. With this device, the FLEX print spooler actually became usable. Previously, the spooler would hang so much during disk accesses as to be useless. The buffer allowed very quick disk access and the process was almost transparent - a 1000% improvement.

Well, the folks at ACS were working on the motherboard at that time and had just gotten some into

stock. They had designed and laid out the whole board themselves. I was impressed with the thickness of the glass as well as the quality of the copper and plating. The board was about twice the thickness of the MP-B2 and had the extended address lines of the SS-50 bus freed from the baud rates. The baud rate generator was still a MC-14411, but it was now on the motherboard, attached and buffered into the SS-30 lines only. Also their selection of rates matched the basic rates used by SWTPC on the newer motherboards they were using in the S09 and S+ systems. The baud rates available are 110, 300, 1200, 4800 and 9600. The 300 and 1200 rates I use for modem communications and the 9600 is used for the main terminal on the system.

Merle and Ross let me build my own board, but they did teach me a technique of assembling that they are using when they are selling boards "factory" wired. If you are in the market for a new motherboard, you may want to consider buying an assembled unit from them, as they will assemble it using this technique. First of all, they are using SQUARE molex pins rather than the round ones usually found in the older motherboards. These make much better board contact. I have shown the motherboard to several people and they have all been impressed with the positive contact these pins afford. The magic of ACS's method lies however, in using 2 motherboards together to assemble the pins into the various slots. They place the pins into one of the boards and use another one to force a perfect alignment before soldering. When the board has been soldered, and the other board removed, the pins are almost perfectly aligned. It is remarkable to see all the cards standing straight up for the first time. Most of my cards had a slight list to them when installed in the older motherboard (ales, my Motorola-SWTPC 64K memory card, which was cut too long at the bottom, making the molex receptacles stretch, still leans forward).

Another advantage to this motherboard, which really was a design effort on ACS's part to accommodate their NMI switch on the left side of their computer system, is that the first slot of the SS-50 slots is offset to the right. Anyone using an older SWTPC mainframe with a DMAF2 disk controller knows that the left edge of the card ends up forced into the chassis brace of the mainframe. You either learn to live with this or cut out the brace which decreases the rigidity of the structure. The DMAF2 card, used in this first slot will now almost perfectly align with the other cards on the right edge allowing a perfect fit.

ACS also showed me a very clever mounting method that they are using in their own main frames. Years ago, I decided the the SWTPC supplied molex board supports were unsatisfactory. Removing a card from the bus almost always entails pulling the motherboard out of the chassis and sometimes breakage of the supports. I had decided to use screws and spacers under the board with nuts above the board to permanently mount the motherboard in the chassis. This worked much better than the factory molex supports as you could now remove and replace cards easily. ACS uses a similar method, but, the have 6-32 x 3/4 screws installed in the chassis and (clever) a spacer which is hex shaped and threaded. You install the screws from the bottom, use the spacers to permanently mount the assembly into the chassis, place the mother board on top and use another set of these spacers to retain the motherboard. Since they are hex shaped, they are easily removed and installed by a simple 1/4 inch nut driver. The areas where these supports are on the motherboard are all ground plane and the solder mask does not cover these areas so that the chassis is well grounded to the motherboard. The DMAF2 card and disk connector were also too tall to allow the use of the cover properly with the older motherboard. This fault is also corrected using the ACS mounting system.

The board is very well solder masked, and as a result easily assembled. The plated throughs are among the best. I still tend to fill plated throughs with solder to reduce tension on boards when they are removed and replaced. Plated throughs on memory cards bother me especially.

The particulars on this board are: .093" thick glass epoxy board with 2oz copper and all feed-throughs plated, 8 - SS-50 pin slots, 8 - SS-30 pin slots, Baud rate generator on board. The baud rate generator is buffered by a 74L04 before going to the buss. All logic power on the mother board is supplied by a 7805 regulator. The data buss is driven by a 74LS640. 16-byte port address decoding is switch selected on any 128 byte boundary from \$8000 to \$F000. The decoding for this is supplied by a 74LS138 and a 74LS32. Individual ports in this space are decoded by another 74LS138. A0-A3 are

sent to the IO by a 74LS367 or 8T97 buffer. There are also several spare gates on board. A 74LS368 buffers RST, R/W, and E. The IRQ and FIRO lines have 10k pullups installed. All of the chip locations on the board are clearly marked for the chip name (not things like "U1" or "U2" but the actual name of the chip like "74LS138") and the pin one locations have a dot to indicate their direction.

The 50 pin buss also has 3 spare lines on either side of the 50 pin lines, for a total of 6 extra user-defined lines. These are very handy for battery backup cards for one thing. These six lines are brought to the rear of the card, near the seventh slot of the 30 pin buss, this area can be used for termination, breadboarding, etc. These six lines have been pre-defined on the motherboard as "x1, bat, x2" on the right and "x3, x4, x5" on the left side. The front of the motherboard has power, ground and reset hookups similar to the ones found on the older motherboards and I made up a harness exactly like the one on the original motherboard for plugging into the power supply molex connectors. This even allows me to change the motherboards quickly if I want. The rear of the ACS motherboard has a series of connections intended for use in the ACS "stacking" computer. ACS refers to this as a "VP-20" buss. The buss has the 8-volt, +16-volt, -16-volt and ground connections, as well as the 6 user-defined lines with the "bat" connections all set up for battery backup. The 30 pin buss is brought out to this buss with 18 lines plus one user-defined line, for easy expansion of the IO section. There are also two "D" lines left spare.

Each port will decode 16 address so you can use cards with multiple IO ports. Hazelwood offers a 6850 4-port card. SWTPC also has a 4-port serial card but it runs a different ACIA and must be programmed differently than the 6850. AAA, of Chicago, is offering a 2 - port serial card. Older MP-S cards from SWTPC will work but show up as four cards in one slot. This means that \$E000, \$E004, \$E008, \$E00C are all the same card. The MP-S could be modified to give only one address decode, but why? It still is one port taking up a 4-port address space. Just be careful when programming. Incidentally, UniFLEX uses \$E000 for output, FLEX uses \$E004 and OS-9 uses \$E004, if you could run all three on the same system, you would never have to move an MP-S card around (a dubious advantage).

The motherboard was fitted into my old SWTPC chassis (which STILL says "6800" on the front), and fits perfectly in that space. I am not sure, but I don't know of another motherboard, which has the same amount of board and port space, that isn't larger than the old chassis. The new motherboard, like the PB4 port buffer has exceeded my expectations. Yes, you can get cheaper boards but, the quality construction of these units and the careful design and execution are worth the extra. This unit is being used in our facilities for both demonstration of operating systems and for repair convenience. I can pull the processor card and the DMAF2 card out and switch rapidly between FLEX and OS-9. We are also using this mainframe to test and burn-in memory cards. This also involves removing and replacing cards quite often. This motherboard, with its square pins and extreme rigidity has supplied an ideal tool for us. I rate it AAA.

Roger Abrahams
Quest Computers
4680 W. Bradley Road
Brown Deer, Wisconsin
53223

(414) 355-4303

CCSM STANDARD MUMPS REVIEW

The version of ANSI standard MUMPS written by Computer Consultants for the 6809 is an interpreted language packaged with special multi-tasking

operating system, and a set of utility programs. The language is not very good for elegant structured programming, but it uses an unusually useful file structure.

MUMPS (Massachusetts General Hospital's Utility MultiProgramming System) was developed by the Laboratory of Computer Science at Massachusetts General Hospital. The design goal was an interactive language running in a time-shared environment on a minicomputer that had strong support for text strings in its file system. Development started in 1967, and must have been finished within a few years because in 1972 there were at least fourteen different versions of the language. The ANSI standard definition of MUMPS was accepted in 1977. There is a new standard for MUMPS in the works, which CCSM MUMPS implements.

Some versions of MUMPS run under a normal operating system, but CCSM MUMPS comes integrated with its own operating system. It cannot be run under OS-9 or FLEX, nor can any FLEX or OS-9 program be run under the MUMPS operating system. This is not as bad as it sounds. It is very much like being stuck in a version of BASIC with lots of BASIC utilities available.

The MUMPS File System

The foundation which CCSM MUMPS is built on is a B-Tree file system. A B-Tree is the most highly regarded way to index data on a disk. In MUMPS, data files, programs, and temporary storage are all stored in this efficient structure, but appear to be in memory. The programmer never has to think about reading or writing the disk. Data files are called global variables. A MUMPS program may believe that the computer has a tremendous memory even if there is only 56K of real memory. The system is essentially using the same virtual memory tricks used in larger systems. MUMPS files are manipulated as if they were a strange type of array. It is, for example, valid to say:

SET @A(1,1) = @A(1,15) where @A denotes a global variable. To do the same thing in a more conventional language would require at least a explicit read and write.

The new (not yet standard) level of MUMPS, allows subscripts to be strings instead of just numbers. This makes statements like:

SET @LIB(Peter,gender) = "Male" possible. The B-Tree structure makes it possible to find the member of the array (file) with subscripts of "Peter", "gender" in a reasonably short time. It is also possible to expand the data file with no particular effort. I could add @LIB(Peter,PlantCount) to the file by just assigning it a value.

MUMPS stores all information on disk as variable length strings. This wastes a little space, but makes the disk structure of the file independent of the lengths of fields. Programmers using MUMPS are most likely not at all worried about the new longer zip code. If a program doesn't check for validity of the zip code field by insisting on five numeric characters, the field can be lengthened without programmer intervention.

The MUMPS Language

It is futile to try to explain an entire language in one short review, especially when there is an operating system to talk about as well. MUMPS is a language which is easy to learn superficially, but full of subtle ways to do things which give a surprising amount of depth. I learned it just well enough to determine that the CCSM implementation

works, and to pick up the flavor of the language.

The standard MUMPS commands are:

- BREAK -- Await a signal
- CLOSE -- Release a device
- DO -- Like Call or Gosub (no arguments)
- ELSE -- Usual meaning
- FOR -- Much extended from the familiar FOR
- GOTO -- Just what you expect
- HALT -- Terminate current process
- HANG -- Pause for a number of seconds
- IF -- Does the usual stuff. Only controls one

line

- KILL -- Free storage allocated to variables
- LOCK -- Lock, or unlock a resource.
- OPEN -- Obtain exclusive ownership of a device
- QUIT -- Define exit point for FOR, DO, or

EXECUTE

- READ -- Input from the current device
- SET -- needed for assignment statements
- USE -- designate the "current device"
- VIEW -- Implementation dependent
- WRITE -- Output to the current device.
- XECUTE -- Interpret and execute arguments
- Z -- extensions to the standard

The standard MUMPS functions are:

- \$ASCII -- Takes the ORD of one character in a string
- \$CHAR -- Translates a series of integers to characters
- \$DATA -- Finds out if a variable is defined
- \$EXTRACT -- A typical Substring function
- \$FIND -- Return the position of a substring in a string
- \$JUSTIFY -- Right justify a string or number
- \$LENGTH -- Return the number of characters in a string
- \$NEXT -- Value of the next subscript in an array
- \$PIECE -- Picks a substring out from between two delimiting substrings.
- \$RANDOM -- Generate a random number
- \$SELECT -- Evaluates a series of pairs of expressions.
 - The first in each pair is evaluated for a boolean true/false. When the first true value is found, the function ends and returns the value of the second expression in that pair.
- \$TEXT -- Returns the text of a line of code.
- \$VIEW -- Implementation-specific
- \$Z -- Non-standard functions start with \$Z

A powerful feature of MUMPS that isn't evident from the list of commands is the pattern match-string operator. The expression VAR?pattern is true if the value of VAR matches the pattern -- VAR?'pattern' is true if VAR doesn't match. The pattern can contain specific strings that must be matched, and general specifications:

- A -- Matches Alphabetics
- C -- Matches control characters
- E -- Matches everything
- L -- Matches lower case alphabetics
- N -- Matches numerics
- P -- Matches punctuation
- U -- Matches upper case alphabetics

The number of times a specification should be matched (called the multiplier) is placed before the specification. For example, '1P5A1' would match a string that consisted of a punctuation mark, followed by five alphabetic characters, followed by a period. If you want to match any number of something, a period can be used as the multiplier. Specifications can be concatenated (AP would match alphabetics and punctuation).

The extensions to the ANSI standard implemented

In CCSM MUMPS are:

The JOB command, which starts a new task, and the \$ORDER function which works about like \$NEXT, but is intended to replace it.

The \$VIEW function is used to examine real memory. The VIEW command is used to change bytes in memory. Two of the Z commands are especially interesting: ZONLINE connects the terminal to a specified device. This is intended to allow the terminal to be attached to a modem. ZEXECUTE lets a MUMPS program execute assembly language routines.

CCSM MUMPS Utilities

CCSM MUMPS comes with a set of utility programs. Since MUMPS is a widespread language with a mature base of users (some of them very large), there are sources for MUMPS programs, including more utilities.

The full screen editor utility only works if you have just the right hardware. I don't.

There is an online help utility called ZH which prints help lines from programs, or sections from the online help manual.

The ZI utility is a great help in finding a missing routine. It lists the first three lines in each file that matches a given search criterion.

The ZP command is a directory command. It gives a list of the files on the current device.

The ZZ command is a line editor. It is about like most other medium good line editors.

There are a family of utilities available from a special menu. The are:

- Edit groups of routines
- List routines by groups
- Copy routines across drive
- Kill routines
- Change/search for strings in routines
- List global structures (data files) and contents
- Copy globals
- Pack a structure (group of files)
- Print Diskette name and volume number
- Change diskette name and volume number
- Copy Diskette
- Reset time and date
- Copy MUMPS OS
- Format new diskette
- Print free blocks

Sample of MUMPS programming

David Brown was kind enough to let me print one of the sample programs from a book on MUMPS that has written (Cookbook of MUMPS published by Eclectic Systems). I considered expanding the program from the compressed form that he (like most MUMPS programmers) writes in, but I think the program makes the point that MUMPS is a very compact and powerful language best as it stands.

There are a few additional facts about MUMPS that will help the most determined readers read this program. All MUMPS commands can be abbreviated to the minimum length that is unambiguous. For example, WRITE can be abbreviated W, and IF can be written I. In a WRITE statement '!' means skip to the next line. A caret '^' before a variable name means the variable is global. More than one command can be placed on a line. Finally, if a command is followed by a colon and a boolean expression (Q:X='') the command will be executed if the boolean value is true.

The following program presents a menu giving a choice of initializing a list of names, adding names to the list, or printing the list. If you decide to enter names, it will prompt for a name; make certain that it has valid form and that it is not already in the name file; and add it to the file. Since the file is kept in a B-Tree structure, the file can be read randomly by the index value, name, or sequentially using the \$ORDER function. B-Tree files are inherently sorted, so the file will be printed out in sorted order by just running through the file using \$ORDER.

```
LISTY ; DBB,,,BOOK;7MAY82 2:32PM;ALPHA LIST EXAMPLE;
LIST M !,"OPTION:",!
  M "1. INITIALIZE",!, "2. ENTER NAMES",!,
    "3. PRINT LIST",!
  R ?5,":","OPT,!, J OPT=" K OPT,X Q
  I OPT?'2IN!(OPT<1)!(OPT>3) M #7,"PLEASE CHOOSE 1-3",
    ! G LIST
  D INIT:OPT=1,ENTER:OPT=2,PRINT:OPT=3 G LIST
INIT S X="Y" I $D(^ALPHA) M #7,"ARE YOU SURE ? " R X,!
  I $E(X)="Y" K ^ALPHA
  E M "NOT "
  M "INITIALIZED !",! Q
ENTER R "NAME (LAST, FIRST MI):",X,! Q:X=""
  I X?'1.AI*,^IA.AP M #7," BAD FORMAT",! G ENTER
  I $D(^ALPHA(X)) M #7," THIS NAME ALREADY EXISTS",!
    G ENTER
  S ^ALPHA(X) = "" G ENTER
PRINT M !," S X="
PRNT2 S X=$D(^ALPHA(X)) Q:X=""
  M X,! G PRNT2
```

Limitations

The MUMPS language is hard to criticize. For what it is, it is very good. It doesn't have enough restrictions to make a good structured programming language, and there is no way to pass parameters to procedures when they are called. I am fascinated by MUMPS, but I don't approve of it.

For many business applications, particularly simple database problems, MUMPS would be excellent, but for other types of processing it would be very poor. MUMPS may be the worst language I know for scientific calculations. It is interpreted, therefore slow, and is missing all the predefined mathematical functions one might want.

The worst problem with CCSM MUMPS is that it is more than just a language. As an operating system, it is too limiting. I had to reconfigure my hardware every time I wanted to test MUMPS, chiefly because it can only deal with five and a quarter inch floppy disk drives and hard disks -- I have my hardware configured to boot off of an eight inch drive. There isn't a documented way for me to add a new device driver to the system, or even an assembler to write the driver in. With the right hardware, it should be possible to make MUMPS the alternate operating system in a GIMIX software switching system. If I didn't have to run FLEX every few months, I might make MUMPS into my second system, and live with a five and a quarter inch system drive.

Summary

I have seen MUMPS running on a DEC and a Data General minicomputer. I don't know how well MUMPS

ran on those machines, but both were running commercial programs of which simple database operations were an important part. Both versions of MUMPS were said to be ANSI Standard. There is a lot to be said for using a standard language, particularly when you are using a small computer. When you outgrow your 6809, you can move on to a larger computer without having to rewrite your software.

It is a tremendous advantage to be able to write programs without having to worry about memory. In MUMPS the entire disk appears to be available memory, i.e., there is no need to ever think about memory. The system keeps the most recently referenced pages of memory in main memory, and writes less recently referenced pages out to disk if it needs to find space for a new page from disk. This is the principle behind virtual memory, which is usually implemented in hardware. MUMPS, being an interpreter, doesn't have to rely on hardware to do its paging.

The CCSM MUMPS operating system knows about the DAT on the CPU boards it supports. It uses the DAT to get at all of the memory in the computer with equal ease, not just some selected 64K.

If I could run MUMPS under OS-9, I would frequently find uses for it. There are many times when its slow speed would not hurt, and its simple string handling and database system would be great advantage. Recently I thought of suggesting MUMPS as the best language to write the software for a system that interacts with id-card readers, and opens doors when an authorized card is read. The software for that system would probably have been written more quickly, and be more functional, if it were written in MUMPS.

I am not going to switch from OS-9 to MUMPS. I stopped writing business software years ago, and, as I see it, that is the environment where MUMPS would be of the most use. I will, however, keep the manuals available. I have many friends who frequently have to solve problems that MUMPS would whiz right through.

By: Peter Dibble

CONVERTING FLEX to OS-9

The Conversion of Assembler-Language
Motorola 6809 Application Programs
from the FLEX Operating System
to the OS/9 Operating System

by E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants, Inc.
1454 Latta Lane, Conway, CA 95027
Telephone Number 404-483-1717/4570

INTRODUCTION

This article provides a set of guidelines, procedures, and concepts for the specific purpose of the conversion of assembler-language application programs written for the FLEX operating system to operate under control of the OS/9 operating system.

This article does not provide a comparison of the relative powers or flexibilities of the two systems, but suggests how to use the facilities of OS/9 for the purpose of performing essentially all of the facilities of FLEX. It features the power of OS/9 as applied to the task of simulating a FLEX-style program interface to OS/9.

This conversion is of interest since much current application software written for FLEX must be rewritten for OS/9 for future use, since OS/9 is becoming available on more computers, whereas the growth of FLEX has recently become far slower than that of OS/9. OS/9 and program development aids such as BASIC90, Microvare C, PASCAL, COBOL, are available to assist in the development of new systems for OS/9, but older systems must be converted.

The conversion process is discussed primarily with respect to the functions of the standard FLEX entry points, storage locations, and facilities, and the corresponding OS/9 system requests and facilities. Physical media conversion is also discussed.

The additional attributes required of OS/9 programs, such as module organization and position-independence, are also discussed, as is efficient use of OS/9 system calls and facilities.

This article does not specifically provide a tutorial discussion of FLEX, OS/9, and their corresponding assemblers. Reference manuals describing each of these products may be obtained from their manufacturers.

However, since there is no good single defining document for all of the FLEX entry points, storage locations, and facilities, this article presents definitions for all such external FLEX attributes and their relation to application programs currently intended to run under the FLEX operating system.

The restriction to application programs is intended to exclude programs (such as printer drivers) which are intimately related to the FLEX system itself, and must therefore be essentially rewritten to convert their function to another operating system.

A second article in this series will discuss the conversion of UNIFLEX programs to OS/9.

OS/9 BACKGROUND

The OS/9 operating system is one of several products of Microvare Systems Corp.

It is actually a generic name for two operating systems for computer systems based upon the Motorola 6809 microprocessor. Both versions support multi-user and multi-tasking access. Version 1 of OS/9 supports up to 64K bytes of main memory, whereas version 2 supports up to 1M bytes of main memory.

A smaller amount of application software is currently available for OS/9 than is available for FLEX, although more is becoming available.

OS/9's primary advantages over FLEX lie in the areas of security, multi-access, expandability, and ease of interfacing to new devices requiring complex processing. OS/9 is almost totally interrupt driven.

OS/9 supports a file security system providing for basic control of file and directory access and update. However, assembler language programs may circumvent the security system (with some amount of difficulty).

FLEX BACKGROUND

The FLEX operating system is one of several products of Technical Systems Consultants, Inc. Another is UNIFLEX, discussed later in this series of articles.

At one time, it was the most popular operating system for systems based upon the Motorola 6800, and then the 6809, microprocessors. It is simple to use, inexpensive, reliable, and supports single user access quite well. It supports 64K bytes of main memory directly.

Because of its past and current popularity, a large amount of application software has been written for systems using FLEX.

FLEX supports interrupt handling directly only to the extent that interrupts are used in the printer spooling logic. However, it does not prohibit programs running under its control from supporting interrupt processing themselves, as long as the interrupt processing routines do not conflict with each other.

FLEX provides a rudimentary form of file security for file and catalog access and update. However, assembler language programs may circumvent the security system fairly readily.

OS/9 COMMAND LINE

The OS/9 command line is formatted as follows:

```
command [p1]...[pn] [<sl> [>so] [>eo] [lp] [pe] [A] [;]...I
```

where: command is the name of the program to be executed,
p1, pn are command parameters,
<sl> is a redirected standard input path,
>so is a redirected standard output path,
>eo is a redirected standard error output path,
lp, pe is an indicator specifying pipeline processing,
A is an indicator specifying background processing,
[;]...I is multiple commands, executed left to right.

In addition, OS/9 supports the user entry of CTRL-C to kill the current task running from the terminal and of CTRL-A to repeat the last line of input from the terminal, for further editing.

FLEX COMMAND LINE

The FLEX command line has the following format:

```
cl[;]...cln command [p1]...[pn] [;]...I
```

where: cl-cln are pre-commands (such as printer drivers, etc.),
command is the name of the program to be executed,
p1-pn are command parameters,
[;]...I is multiple commands, executed left to right.

OS/9 PATH NAMES

An OS/9 path name has the following format:

```
/|dir1|...|dirn|/name
```

where: /dir1.../dirn/ is the specification of the path,
name is a 1-29 character file name, starting with a letter,

Program files default to the execution path (set by the "chx" command) and data files default to the data path (set by the "chd" command).

FLEX FILE NAMES

A FLEX file name has the following format:

```
[d] [v] [name] [suffix]
```

where: drive is drive number 0-3;
name is a 1-8 character file name, starting with a letter,
suffix is a 1-3 character suffix.

Program files default to the system drive and data files default to the work drive (both set by the "asn" utility command).

OS/9 PROGRAM INTERFACE SUMMARY

Every OS/9 application program is composed of one or more OS/9 modules. An OS/9 module is known to the system by possessing a properly-formatted module header and trailer, both of which are described later in this article.

Application programs request services from OS/9 in the following simple manner: each request is a system call (specifically, swt2), followed by a one-byte parameter, of the following symbolic format:

os9 code

where "code" represents one of the standard OS/9 system call numbers. The OS/9 assemblers translate this symbolic format into the equivalent of the following object code:

swt2

fcb code

Parameters are passed in registers and in memory areas referenced by the registers between the application modules and OS/9 and vice versa. Application programs may not normally inspect or modify OS/9 system areas. In OS/9 version 2, application programs may not inspect or modify memory locations associated with other users or tasks, since they are not in the addressable memory space.

OS/9 may be easily configured to support at least the following devices, depending upon the implementation:

- 5.25" floppy disks,
- 8" floppy disks,
- hard disks,
- console display,
- console keyboard,
- serial printer,
- parallel printer,
- specialty printers,
- etc.

Files intended for the console may be redirected to any other device. Other devices may be directly supported by application programs on OS/9 Level 1, or by device drivers on OS/9 Level 1 or 2. Device drivers are usually provided for a subset of the devices listed above. Some device parameters, such as the echo flag for the console keyboard, may be modified easily in a device descriptor associated with each device.

Files for all devices are handled in a very similar manner by OS/9. The I/O subsystem is invoked thru a subset of the OS/9 system calls. All disk files are accessible sequentially and randomly, and no distinction is made between files containing binary arbitrary data and text (printable) data; all disk files may contain arbitrary contents. The logical block size on disk and similar devices is 256 bytes.

An application program may create additional tasks and specify that they are to run in asynchronous, synchronous, or planned mode, and, in the latter two modes, monitor their progress and completion status.

Since modules for OS/9 must be written to use pure position-independent code, they may be loaded as required in the logical address space, and multiple users of the same module share the same code, although each user has an independent data address space. Programs must be loaded into contiguous memory addresses, but OS/9 handles the memory allocation.

FLEX PROGRAM INTERFACE SUMMARY

Application programs request services from FLEX in a simple manner: each request is a jump to subroutine (JSR), of the following format:

jsr entry

where "entry" represents one of the standard FLEX entry point vectors. Parameters are passed in registers and in memory areas between the application modules and FLEX and vice versa. Application programs may inspect and change FLEX storage locations directly.

FLEX (or, specifically, SWTPC FLEX) directly supports the following devices:

- 5.25" floppy disks,
- 8" floppy disks,
- hard disks,
- console display,
- console keyboard,
- serial printer,
- parallel printer,
- specialty printers,
- etc.

Only four disk drives may be addressed simultaneously, regardless of type. Files intended for the console may be redirected to disk or printer. Other devices may be directly supported by application programs.

The disk file management system (FMS) is simple to invoke by an application program, utilizing a file control block (FCB) for essentially all disk file-related program-FLEX coordination. The FCB sector buffer (and physical disk sector) length is 256 bytes in length and the FCB prefix length is 64 bytes in length, making the total FCB 320 bytes in length.

When a program invokes one of the FMS entry point vectors and passes parameters to it thru an FCB, FMS returns an error indicator in a register and an error code in the FCB. FMS implements both random and sequential files, although they do not have interchangeable access, and both binary files (with arbitrary contents) and text files (with compressed spaces and suppressed control characters).

An application program may call FLEX as a subroutine, passing it a command line, and thus execute another program in a synchronous manner (as long as the memory spaces do not conflict), and receive back a limited amount of information in terms of an FMS return code.

A given application program is always loaded into the same areas of memory, even though it may in reality be position-independent. A program may be loaded into contiguous or discontinuous memory addresses, and handles its own memory allocation.

OS/9 FACILITIES

The OS/9 system and I/O request codes (used in "os9 code" assembler-language statements) are as follows:

OS/9 System Service Request Codes

Code	Name	Description
\$00	\$SLink	Link to Module
\$01	\$SLoad	Load Module from File
\$02	\$SUnlink	Unlink Module
\$03	\$SFork	Start New Process
\$04	\$SWait	Wait for Child Process to Die
\$05	\$SChain	Chain Process to New Module
\$06	\$SExit	Terminate Process
\$07	\$SMem	Set Memory Size
\$08	\$SSend	Send Signal to Process
\$09	\$SIcpt	Set Signal Intercept
\$0A	\$SSleep	Suspend Process
\$0B	\$SSpd	Suspend Process
\$0C	\$SID	Return Process ID
\$0D	\$SPrior	Set Process Priority
\$0E	\$SSWI	Set Software Interrupt
\$0F	\$SPerr	Print Error
\$10	\$SPyMem	Parse Pathlist Name
\$11	\$SCmpMem	Compare Two Names
\$12	\$SChBlt	Search Bit Map
\$13	\$SAlBlt	Allocate in Bit Map
\$14	\$SDelBlt	Deallocate in Bit Map
\$15	\$STime	Get Current Time
\$16	\$STime	Set Current Time
\$17	\$SCRC	Generate CRC
\$18	\$SGPRDesc	Get Process Descriptor Copy
\$19	\$SGBlkMap	Get System Block Map Copy
\$1A	\$SModDir	Get Module Directory Copy
\$1B	\$SEpyMem	Copy External Memory
\$1C	\$SUser	Set User ID number
\$1D	\$SUnLoad	Unlink Module by name

OS/9 System Service Reserved Request Codes

Code	Name	Description
\$28	\$SSRqMem	System Memory Request
\$29	\$SSRtMem	System Memory Return
\$2A	\$SIrq	Enter I/O Polling Table
\$2B	\$SIQov	Enter I/O Queue
\$2C	\$SAIProc	Enter Active Process Queue
\$2D	\$SNProc	Start Next Process
\$2E	\$SVModul	Validate Module
\$2F	\$SFInd64	Find Process/Path Descriptor
\$30	\$SAI164	Allocate Process/Path Descriptor
\$31	\$SRet64	Return Process/Path Descriptor
\$32	\$SSSvc	Service Request Table Initialization
\$33	\$SI0del	Delete I/O Module
\$34	\$SLink	System Link
\$35	\$Sboot	Bootstrap System
\$36	\$SRtMem	Bootstrap Memory Request
\$37	\$SGProcP	Get Process Pointer
\$38	\$SMove	Move Data (Low Bound First)
\$39	\$SAIRAM	Allocate RAM Blocks
\$3A	\$SAIimg	Allocate Image RAM Blocks
\$3B	\$SDelimg	Deallocate Image RAM Blocks
\$3C	\$SSetimg	Set Process DAT Image
\$3D	\$SFreeLH	Get Free Low Block
\$3E	\$SFreeHB	Get Free High Block
\$3F	\$SAITask	Allocate Process Task Number
\$40	\$SDelTask	Deallocate Process Task Number
\$41	\$SSetTask	Set Process Task DAT registers
\$42	\$SResTask	Reserve Task Number
\$43	\$SRelTask	Release Task Number
\$44	\$SDATLog	Convert DAT Block/Offset to logical
\$45	\$SDATmp	Make temporary DAT image
\$46	\$SLDAXY	Load A [X,Y]
\$47	\$SLDAXYP	Load A [X,Y]
\$48	\$SLDDXXY	Load D [D,X,Y]
\$49	\$SLDABX	Load A from D,X in Task B
\$4A	\$SSTARX	Store A at D,X in Task B
\$4B	\$SAIPrc	Allocate Process Descriptor
\$4C	\$SDelPrc	Deallocate Process Descriptor
\$4D	\$SLink	Link using Module Directory Entry
\$4E	\$SFModul	Find Module Directory Entry

OS/9 I/O Service Request Codes

Code	Name	Description
\$80	\$SAttach	Attach I/O Device
\$81	\$SDetach	Detach I/O Device
\$82	\$SDup	Duplicate Path
\$83	\$SCreate	Create New File
\$84	\$SOpen	Open Existing File
\$85	\$SMkdir	Make Directory File
\$86	\$SChgDir	Change Default Directory
\$87	\$SDelete	Delete File
\$88	\$SSeek	Change Current Position
\$89	\$SRead	Read Data
\$8A	\$SWrite	Write Data
\$8B	\$SReadLn	Read Line of ASCII Data
\$8C	\$SWriteLn	Write Line of ASCII Data
\$8D	\$SGetStt	Get Path Status
\$8E	\$SSetStt	Set Path Status
\$8F	\$SClose	Close Path
\$90	\$SDeleteX	Delete from Current Execution Directory

The OS/9 system and I/O request return codes (returned in the B register by OS/9 system routines) are as follows:

OS/9 Error Return Codes

Code	Name	Description
\$00	-----	No Error
\$08	\$SPthFull	Path Table Full
\$09	\$SRPNum	Bad Path Number
\$0A	\$SPoll	Polling Table Full
\$0B	\$SRMode	Bad Mode
\$0C	\$SDevOvf	Device Table Overflow
\$0D	\$SRMID	Bad Module ID
\$0E	\$SDirFull	Module Directory Full
\$0F	\$SMemFull	Process Memory Full
\$10	\$SUnkSvc	Unknown Service Code
\$11	\$SModBsy	Module Busy
\$12	\$SRPAddr	Bad Page Address
\$13	\$SEOF	End of File
\$15	\$SNES	Non-Existing Segment
\$16	\$SFNA	File Not Accessible
\$17	\$SRPNam	Bad Path Name
\$18	\$SPNMF	Path Name Not Found
\$19	\$SSLF	Segment List Full
\$1A	\$SECF	Creating Existing File
\$1B	\$SIRA	Illegal Block Address
\$1D	\$SMNF	Module Not Found
\$1F	\$SDelSP	Deleting Stack Pointer Memory
\$20	\$SIProcID	Illegal Process ID
\$22	\$SNoChild	No Children
\$23	\$SISW	Illegal Srv Code
\$24	\$SPrcAbt	Process Aborted
\$25	\$SPrcFull	Process Table Full
\$26	\$SIForkP	Illegal Fork Parameter
\$27	\$SKwnMod	Known Module
\$28	\$SRMCRC	Bad Module CRC
\$29	\$ESUSlp	Unprocessed Signal Pending
\$2A	\$SNEMod	Non-Existing Module
\$2B	\$SRNam	Bad Name
\$2C	\$SRMHP	Bad Module Header Parity
\$2D	\$SNoRam	No Ram Available
\$2E	\$SRProcID	Bad Process ID
\$2F	\$SNoTask	No Available Task Number
\$30	\$SUnit	Illegal Unit (Drive)
\$31	\$SSect	Bad Sector Number
\$32	\$SWP	Write Protect
\$33	\$SCRC	Read Check Sum
\$34	\$SRred	Read Error
\$35	\$SWrite	Write Error
\$36	\$SNotRdy	Device Not Ready
\$37	\$SSeek	Seek Error
\$38	\$SFull	Media Full
\$39	\$SBtTyp	Bad Type (incompatible) Media
\$3A	\$SDavBsy	Device Busy
\$3B	\$SDIDC	Disk ID Change
\$3C	\$Sack	Record Busy (Locked-Out)
\$3D	\$SShare	Non-shareable File Busy
\$3E	\$SDeadlk	I/O Deadlock Error

The OS/9 direct page variables (accessible to application programs under OS/9 Level 1) are defined as follows:

Direct Page Variables

Address	Name	Description
\$20	D.FMEM	Free memory bit map pointers
\$24	D.MEM	Memory limit
\$26	D.ModDir	Module directory
\$2A	D.Init	Rom base address
\$2C	D.SW13	Sw13 vector
\$2E	D.SW12	Sw12 vector
\$30	D.FIRQ	Firq vector
\$32	D.IRD	Irq vector
\$34	D.SW1	Sw1 vector
\$36	D.NM1	Nm1 vector
\$38	D.SvcIRQ	Interrupt service entry
\$3A	D.Poll	Interrupt polling routine
\$3C	D.UserIRQ	User irq routine
\$3E	D.SysIRQ	System irq routine
\$40	D.UserSvc	User service request routine
\$42	D.SysSvc	System service request routine
\$44	D.UserDis	User service request dispatch table
\$46	D.SysDis	System service request dispatch table
\$48	D.Slice	Process time slice count
\$49	D.PrcDBT	Process descriptor block address
\$4B	D.Prc	Process descriptor address
\$4D	D.AProcQ	Active process queue
\$4F	D.WProcQ	Waiting process queue
\$51	D.SProcQ	Sleeping process queue
\$53	D.Time	Time (YMMDDHHMMSSIT)
\$55	D.Year	Year in century
\$54	D.Month	Month in year
\$55	D.Day	Day in month
\$56	D.Hour	Hour in day
\$57	D.Min	Minute in hour
\$58	D.Sec	Second in minute
\$59	D.Tick	Tick in second
\$5A	D.T	Ticks per second
\$5B	D.TSlice	Ticks per time-slice
\$5C	D.IORL	I/O manager free memory low bound
\$5D	D.IORH	I/O manager free memory high bound
\$5E	D.DevTbl	Device driver table address
\$5F	D.PollTbl	Irq polling table address
\$60	D.PYHDBT	Path descriptor block table pointer
\$61	D.BTLO	Bootstrap low address
\$62	D.BTHI	Bootstrap high address
\$63	D.DMAReq	DMA in use flag
\$64	D.AltIRQ	Alternate irq vector
\$65	D.KbdSta	Keyboard scanner static storage
\$66	D.DiskTmr	Disk motor timer
\$67	D.Clock	Address of clock tick routine

The OS/9 device and file descriptor offsets are as follows:

OS/9 Random Block Path Descriptor Format

\$0A PD.SMF	State flags
\$0B PD.OP	Current logical byte position
\$0F PD.SIZ	File size
\$13 PD.SBL	Segment beginning lsn
\$16 PD.SBP	Segment beginning psn
\$19 PD.SSZ	Segment size
\$1C PD.DSK	Disk id
\$1E PD.DTB	Drive table ptr
\$20 PD.DEV	Device type
\$21 PD.DRV	Drive number
\$22 PD.STP	Step rate
\$23 PD.TYP	Disk device type (5", 8", other)
\$24 PD.DMS	Density capability
\$25 PD.CTL	Number of cylinders
\$27 PD.SID	Number of surfaces
\$28 PD.VFY	Verify disk writes
\$29 PD.SCT	Default sectors/track
\$2B PD.TOS	Default sectors/track track zero
\$2D PD.ILV	Sector interleave offset
\$2E PD.SAS	Segment allocation size
\$2F PD.TFM	DMA transfer mode
\$30 PD.Exttn	Path extension for record locking
\$33 PD.ATT	File attributes
\$34 PD.FD	File descriptor psn
\$37 PD.BFD	Directory file descriptor psn
\$3A PD.DCP	File directory entry pointer
\$3E PD.DVT	User readable device table pointer

OS/9 Random Block Path Extension Format

\$00 PE-PE	PE path number
\$01 PE.PDptr	Back pointer to path descriptor
\$03 PE.MFPI	Drive open-file list pointer
\$05 PE.Confl	Circular file conflict list
\$07 PE.Lock	Path lockout status
\$08 PE.LoLck	Low locked logical address
\$0C PE.HiLck	High Locked Logical address
\$10 PE.Wait	PE pointer to next locked-out PE
\$12 PE.InOut	Max ticks to wait for locked segment
\$14 PE.Owner	Process ID of owner of locked segment

OS/9 Device Descriptor Format

\$00 DD.TOT	Total number of sectors
\$03 DD.TKS	Track size in sectors
\$04 DD.MAP	Number of bytes in allocation bit map
\$06 DD.BIT	Number of sectors/bit
\$8 DD.DIR	Address of root directory file descriptor
\$9 DD.OWN	Owner
\$00 DD.ATT	Attributes
\$0E DD.DSK	Disk id
\$10 DD.FMT	Disk format: density/slides
\$11 DD.SPT	Sectors/track
\$13 DD.RES	Reserved for future use
\$15 DD.SIZ	Device descriptor minimum size
\$16 DD.BT	System bootstrap sector
\$18 DD.BSZ	Size of system bootstrap
\$1A DD.DAT	Creation date
\$1F DD.NAM	Volume name
\$3F DD.OPT	Option area

OS/9 File Descriptor Format

\$00 FD.ATT	Attributes
\$01 FD.OWN	Owner
\$03 FD.DAT	Date last modified
\$06 FD.LNK	Link count
\$09 FD.SIZ	File size
\$0D FD.OrExt	Segment list extension
\$10 FD.SEG	Beginning of segment list

OS/9 Directory Entry Format

\$00 DIR.NM	File name
\$10 DIR.FD	File descriptor physical sector number
\$20 DIR.SZ	Directory record size

OS/9 File Descriptor Offsets

\$0A PD.OV2	Output Device Table Pointer
\$0C PD.RAW	Rread/Write or Rdlin/Wrlin Mode
\$0D PD.MAX	Rreadline High Byte Count
\$0F PD.MIN	Devices Are Mine If Clear
\$10 PD.STS	Status Routine Module Address
\$12 PD.S7M	Reserved for Status routine
\$20 PD.DEVT	Device type
\$21 PD.UPC	Case (0=both, 1=upper)
\$22 PD.BSO	Backsp (0=bse, 1=bse,sp,bse)
\$23 PD.DLO	Delete (0=bse over line, 1=crlf)
\$24 PD.EKD	Echo (0=no echo)
\$25 PD.ALF	Autolf (0=no auto lf)
\$26 PD.NUL	End of line null count
\$27 PD.PAU	Pause (0=no end of page pause)
\$28 PD.PAG	Lines per page
\$29 PD.BSP	Backspace character
\$2A PD.DEL	Delete line character
\$2B PD.EOR	End of record character
\$2C PD.EOF	End of file character
\$2D PD.RPR	Reprint line character

\$2E PD.DUP	Duplicate last line character
\$2F PD.PSC	Pause character
\$30 PD.INT	Keyboard interrupt character
\$31 PD.QIT	Keyboard quit character
\$32 PD.BSE	Backspace echo character
\$33 PD.OVF	Line overflow character
\$34 PD.PAR	Parity code
\$35 PD.BAU	ACIA baud rate for Color Computer
\$36 PD.DZP	Offset of device name
\$3B PD.XON	ACIA x-on character
\$39 PD.XOFF	ACIA x-off character
\$3A PD.ERR	Most recent I/O error status
\$3B PD.TBL	Device table address (copy)

These, and many other OS/9 symbols, are defined in several standard library files in the "defs" directory. Some of these definition files are as follows:

```

defs/os9defs.2
defs/os9sysdefs.11
defs/os9lodels.1
defs/os9bldels.2
defs/os9scldels.1
defs/ll.aquates
defs/systype
defs/sysdefs.sys

```

Not only do these definition files describe the request codes and error messages; they describe the entire environment in which the OS/9 operating systems and all programs running under it operate, including machine parameters, and descriptor offsets. All S/9 modules and almost all other programs running under OS/9 use the standard OS/9 symbols. The only major variation from the standard naming conventions was caused by Microvare, when they renamed many symbols from the old names used in OS/9 versions lower than 1.2 to the symbols currently used. However, they provide cross-reference definition files to allow the alternate use of the old symbol names.

FLEX AND SBUG FACILITIES

The FLEX storage locations of interest to application programs are as follows:

Address Range	Name	Description
\$0000-\$0fff	-----	application program area
\$c000-\$007f	stack	FLEX stack
\$c080-\$c0ff	inbuff	command line buffer
\$c100-\$c6ff	cmdad	utility load area
\$c800-\$c93f	syslcb	system file control block
\$cacD-\$cbff	spflcb	spool file control block
\$cc00	bspcbr	ttyset backspace
\$cc01	delchr	ttyset delete
\$cc02	eofchr	ttyset end of line
\$cc03	depth	ttyset depth count
\$cc04	width	ttyset width count
\$cc05	nulls	ttyset null count
\$cc06	tabchr	ttyset tab
\$cc07	bsechr	ttyset backspace echo
\$cc08	eject	ttyset eject count
\$cc09	pause	ttyset pause control
\$cc0a	escchr	ttyset escape
\$cc0b	s_drn	system drive number
\$cc0c	w_drn	working drive number
\$cc0d	sflag	use system drive flag
\$cc0e	sysmon	system month
\$cc0f	sysday	system day
\$cc10	sysyr	system year
\$cc11	lstfrm	last terminator
\$cc12	usrcmd	user command table
\$cc14	cbulptr	line buffer pointer
\$cc16	escret	escape return
\$cc18	curchr	current character
\$cc19	prevchr	previous character
\$cc1a	curlct	current line number
\$cc1b	loadao	loader address offset
\$cc1d	xlrlflg	transfer flag
\$cc1e	xlradr	transfer address
\$cc20	errtyp	error type
\$cc21	ioflag	special i/o flag
\$cc22	outswt	output switch
\$cc23	inswt	input switch
\$cc24	foaddr	file output address
\$cc26	fiaddr	file input address
\$cc28	docmdf	command flag
\$cc29	curcol	current output column
\$cc2b	memend	memory end
\$cc2d	errvec	error name vector
\$cc2f	filecho	file input echo flag
\$cc30	flag	ins in use flag
\$cc31	curtsk	current task pointer
\$cc33	cputyp	cpu type flag
\$cc34	mode	mode flag
\$cc35	pt rap	reserved printer area pointer
\$cc37	pt len	reserved printer area length
\$cc39	pt dev	printer device address
\$cc43	retadr	docmd return address
\$cc49	ulclfg	upper/lower case flag
\$cc4e	prompt	pointer to prompt string
\$ccc0	plnt	printer initialization
\$ccd0	pterm	printer close routine
\$ccd8	gchk	printer ready check routine
\$cce4	pout	printer output routine
\$ccfc	prclfg	active spooling flag

\$d3e0 dummy rts for reserve memory function
 \$d409 lchase FMS FCB base pointer
 \$d40b lchcur FMS FCB current address
 \$d435 verify FMS verify flag
 \$d436 surtab FMS surtab table

The FLEX service routines maintain the DP, Y, and U registers, but generally modify some or all of the other registers. The FLEX entry vectors of interest to application programs are as follows:

Address	Name	Description
\$c700	sched	spool swi handler
\$c703	loop	spool nothing-to-do loop
\$c706	stersp	spool entry
\$c709	ststest	spool test and swi flag
\$c70c	clrflg	spool clear flag
\$c70f	lentry	spool irq handler
\$cd00	colds	cold start
\$cd03	warm	warm start
\$cd06	reentry	re-entry
\$cd09	lch	basic input character
\$cd0c	lch2	basic input character from console
\$cd0f	outch	basic output character
\$cd12	outch2	basic output character to console
\$cd15	getchr	get character
\$cd18	putchr	put character
\$cd1b	lnbuff	input into line buffer
\$cd1e	prstrng	print string
\$cd21	class	classify character
\$cd24	pcrlf	print cr/lf
\$cd27	nutch	get next buffer character
\$cd2a	rstrio	restore i/o vectors
\$cd2d	getfll	get file specs
\$cd30	load	load binary file
\$cd33	setext	set extension
\$cd36	addbx	add b to x
\$cd39	outdec	output decimal number
\$cd3c	outhex	output hexadecimal number
\$cd3f	reptr	report error
\$cd42	gethex	get hexadecimal number
\$d45	outadr	output hexadecimal address
\$cd48	lndec	input decimal number
\$cd4b	docand	call FLEX as a subroutine
\$cd4e	stat	check console status

\$d3de		intap	input tape vector
\$d3e1		setirq	set irq vector
\$d3e3		clrirq	clear irq vector
\$d3e5		tlrch	console input without echo
\$d3e7		lhandl	irq handler
\$d3e9		swivac	swi vector
\$d3eb		lrqvac	irq vector
\$d3ed		t off	timer off
\$d3ef		tim on	timer on
\$d3f1		timlnt	timer init
\$d3f3		montr	monitor warm start
\$d3f5		tlnt	console init
\$d3f7		tcheck	console check
\$d3f9		toutch	console output
\$d3fb		tlrch	console input with echo
\$d3fd		jmpint	FLEX initialization

\$d400	fmsini	FMS initialization
\$d403	fmscls	FMS close files
\$d406	fms	FMS call

\$de00	dread	basic read disk
\$de03	dwrite	basic write disk
\$de06	dverify	basic verify disk
\$de09	drest	basic restore disk
\$de0c	drlve	basic select drive
\$de0e	dcheck	basic check drive ready
\$de12	dsulch	basic quick drive check
\$de15	dcolds	basic driver cold start
\$de18	dswarm	basic driver warm start
\$de1b	dseek	basic drive seek-to-sector

The SBUG locations of interest to application programs are as follows:

Addresses	Name	Description
\$d8e0	user-v	user interrupt vector
\$d8e2	sw13-v	sw13 interrupt vector
\$d8e4	sw12-v	sw12 interrupt vector
\$d8e6	flrq-v	flrq interrupt vector
\$d8e8	lrq-v	lrq interrupt vector
\$d8ea	sw1-v	sw1 interrupt vector
\$d8ec	src-up	sw2 vector origin address
\$d8ee	src-v1	sw2 vector limit address
\$dfd0-\$dfd1	datmap	dat map copy

The SBUG entry vectors of interest to application programs are as follows:

Address	Name	Description
\$f800		monitor monitor cold start
\$f802		nextcmd monitor warm start
\$f804		lch get character from console
\$f806		lch2 get character from console and echo
\$f808		lchchk check for input from console
\$f80a		ovtch display character on console
\$f80c		pdstr display character string to \$04
\$f80e		pcrlf display carriage return/line feed
\$f810		prstrng display cr/lf, then display string to \$04

\$f812 || lra load real 20-bit address of memory location

Note that most of the vector entry points start with a byte containing a 6809 "jmp" instruction (\$7e), so that they may be invoked with a "jmp" or "jsr" instruction in the application program. The exceptions are marked with "||" in the table above. These entry points are usually invoked with indirect addressing (i.e., jsr \$f805). Occasionally the other entry points are invoked indirectly, by using the address in the entry point without the preceding "jmp" (at the stated address plus one).

The SBUG entry point vectors and storage locations are also presented above, since many application programs invoke the SBUG facilities directly, rather than using them thru FLEX, for various reasons, such as to avoid the input and output redirection caused by the "in", "out", and "pr" pre-commands.

As described earlier, FLEX performs all disk I/O thru the File Management System (FMS). The requestor indicates the action to be performed by placing a code into the FMS function code of the corresponding FCB. The File Management System function codes are as follows:

Code	Description
\$00	get/put next byte
\$01	open-input
\$02	open-output
\$03	open-update
\$04	close
\$05	rewind
\$06	open directory
\$07	get information record
\$08	put information record
\$09	read single sector
\$0a	write single sector
\$0b	extend directory
\$0c	delete
\$0d	rename
\$0f	get next sequential sector
\$10	open system information record
\$11	get random byte from sector
\$12	put random byte into sector
\$13	open-extend
\$14	find next drive
\$15	position to record n
\$16	back up one record

The requestor places one of these codes into the FCB function code before invoking FMS. The FMS file control block (FCB) has the following format:

Offset	Description
\$00	function code
\$01	error status
\$02	activity status
\$03	drive number
\$04-\$0b	name
\$0c-\$0e	extension
\$0f	file attributes
\$11-\$12	starting disk address
\$13-\$14	ending disk address
\$15-\$16	file size
\$17	file sector map indicator
\$18	assigned drive number
\$19	creation month
\$1a	creation day
\$1b	creation year
\$1c-\$1d	list pointer
\$1e-\$1f	current position
\$20-\$21	current record number
\$22	date index
\$23	random index
\$24-\$2e	name work buffer
\$2f-\$31	current directory address
\$32-\$35	first deleted directory pointer
\$36-\$3a	rename work area
\$3b	space compression flag
\$40-\$4f	sector buffer

When a program invokes the FMS and passes parameters to it thru an FCB, FMS returns an error indicator (the carry flag) and an error code in the same FCB. The FMS error codes are as follows:

Number	Description
00	no error
01	illegal FMS function code
02	the requested file is in use
03	the specified file already exists
04	the specified file count not be found
05	system directory error-reboot system
06	the system directory space is full
07	all available disk space has been used
08	read past end of file
09	disk file read error
10	disk file write error
11	the file or disk is write protected
12	the file is protected-file not deleted
13	illegal file control block specified
14	illegal disk address encountered
15	an illegal drive number was specified
16	drives not ready
17	the file is protected-access denied
18	system file status error
19	FMS data index range error
20	FMS inactive-reboot system

```

21 illegal file specification
22 system file close error
23 sector map overflow-disk too segmented
24 nonexistent record number specified
25 record number match error-file damaged
26 command syntax error-retype command
27 command not allowed while printing
28 wrong hardware configuration

```

Most of the FLEX symbols are defined in a standard SYMTPC library file named FLEXLIB. Unfortunately, the symbols are not used in a standard manner by TSC or by any of the FLEX licensees. Many programs ignore any naming conventions and create their own names, or simply refer to the absolute addresses of the FLEX vectors and storage locations directly. The names used above will be used for this discussion, however.

OS/9 PROGRAM REQUIREMENTS

OS/9 requires that all application modules have a module header and a module trailer, and that the program code and data possess the attribute of position-independence. The OS/9 assemblers generate the proper formats and contents for the module header and trailer, and warn the programmer in the case of non-position-independent construct usage, but it is the programmer's responsibility to ensure the correctness of the position-independence aspects of the program.

The same format is required for memory and disk representations of executable OS/9 modules. OS/9 modules are required to be contiguous on disk and in memory, although an OS/9 program may be composed of more than one OS/9 executable module. Each module is assembled relative to a base address of zero, but is loaded at whatever addresses that OS/9 determines still contain the module.

The OS/9 module header for executable modules has the following format:

Offset	Name	Description
\$00-\$01	M\$ID	ID code of \$87cd
\$02-\$03	M\$Size	Module size
\$04-\$05	M\$Name	Module name offset
\$06	M\$Type	Type and language
\$07	M\$Revs	Attribut s and revision level
\$08	M\$Parity	Header parity
\$09	M\$IDSize	Module ID size
\$0a-\$0b	M\$Exec	Execution offset
\$0c-\$0d	M\$Store	Data storage size

The module trailer is three bytes in length and its contents is based on a check-sum computation of the entire module, starting with the "\$87cd" in the module header.

Conversion of existing programs to possess position-independent code and data structures may be quite straightforward, primarily involving the changes described below, or it may be exceedingly complex. The complexities usually arise in situations involving heavy use of indexed addressing with sixteen-bit offsets, because of the ambiguities between addresses, offsets, and sixteen-bit data fields.

All OS/9 application program code is required to be written using pure position-independent code and data techniques, as noted earlier. When a data space is allocated to an application program to be executed, OS/9 points the U and DP registers to the beginning of the data space, the S and X registers to the end, the Y register to the end of the end of the parameter area (of which X points to the beginning), and the PC register to the beginning of the module plus the execution offset, as specified in the module header.

Because of the multi-user capability of OS/9 modules, the program code area is almost always shared among all the users of a given module, and thus locations within the program must not be altered once a module is brought into memory. Thus, PC-relative store operations are normally forbidden under OS/9, although the hardware will not usually detect such violations, providing the user the task of writing or modifying programs to obey such restrictions, and of converting many PC-relative storage locations to be logically S-relative, U-relative, X-relative, or Y-relative.

Since the offset between the program load point and the data load point is unknown before a program is executed by a particular user, position-independent references ("S", "U", "X", "Y") must be used. In addition, the application program, not OS/9, is responsible for all initialization in the data space; thus constant information (such as tables) should be placed in the program space and referenced in a position-independent-code ("PCR") manner, or copied to the data space. Program packages such as the Microvare "C" compiler perform this task as a part of initialization.

The simplest manner in which to convert or write position-independent code references is to suffix all direct and extended addresses used in instructions with "PCR" or "U" for all extended address references that are within the program area or data area, respectively. This process is not always performed without problems. Specific problems with the conversion of immediate addresses and table references are discussed below.

Because the OS/9 assemblers attempt to optimize the use of the U-register, and the U-register points to the base of the allocated data area, the data area must be specified before any references to it are made, or the assembler may generate "These Error" messages, since it assumes 16-bit offset addressing on the first pass and may discover 0-bit, 5-bit, or 8-bit offset addressing on the second pass. Similarly, symbols should be defined before use, whenever possible, to prevent similar problems with other registers.

Fixed references external to the program must be absolute addresses under OS/9 Level 1, since the effective addresses, computed using the program counter would vary if the application program were moved to a different location than the assumed zero load point. Fixed external references are generally invalid under OS/9 Level 2, since the I/O, OS/9, and other application program areas are mapped out of the application program's address space.

Rather than "jmp" or "jsr" for PC-relative references, the instructions "bra/ibra" or "bsr/lbsr" must be used. The latter instructions accomplish exactly the same purpose, but are slower and longer than the former. They are used when longer branches are required.

Because OS/9 modules must be contiguous and assembled relative to address zero, the assembler "org" statement may not normally be used. FLEX programs containing "org" statements must be reviewed to determine the effect of removing them; often, the effects are very minor.

For 6800 compatibility, FLEX programs have often been coded using the 6800 mnemonics, such as "ldaa" and "lda a". The OS/9 assemblers will not accept the 6800 mnemonics. Thus FLEX programs being converted to OS/9 must be purged of 6800 mnemonics. This task is easily accomplished with a text editor or a commercial translator.

Addresses used in an immediate context may require much attention to achieve position independence. A load instruction of the following format:

```
ldr #addr
```

where r = s, u, x, or y, and "addr" is not external to the program code nor a non-address constant, should be coded in one of the following formats:

```
leqr addr,pcr    OR    leqr addr,u
```

for position independence. In the case of statements of the following format involving the "r" register:

```
ldd #addr
```

the 6809 has no "lead" instruction, and it must be rewritten using one of the other sixteen-bit registers, without causing side-effects. The following code, which provides equivalent results in the "r" and "cc" registers (but changes no other registers), could be used:

```

pshs x
leqr addr,pcr    OR    leqr addr,u
pshs x
ldd ,s++
puls x

```

Sixteen-bit compare instructions of the following format:

```
cmpw #addr
```

(where "r" and "addr" are as defined above) must be rewritten, since there is no compare effective address instruction. For r = u, x, or y, code of the following format could be used:

```

pshs r,d
leqr (($ffff-addr)+1),pcr    OR    leqr (($ffff-addr)+1),u
tfr r,d
addd $02,s
puls r,d

```

For r = s, code of the following format could be used:

```

pshs u,d
leau ($ffff-addr),pcr    OR    leau ($ffff-addr),u
pshs u
tfr s,d
addd ,s++
puls u,d

```

For r = d, code of the following format could be used:

```

pshs x
leax addr,pcr    OR    leax addr,u
pshs x
cmpd ,s++
puls x

```

The general solution to the problem of conversion of address tables to position-independence is to code the addresses in the table as offsets from some base point, where the base point is chosen in some manner convenient to the situation. Two typical base points are the beginning of the table itself and the beginning of the module, for single-module programs. The primary advantage of using the beginning of the module is that the relative offsets and extended addresses are identical, assuming that the module starts at address zero. The primary advantage of using the beginning of the table is that the beginning address is already loaded in a register when the effective address is being calculated, saving one instruction.

For example, consider the following program fragments:

```

:
: ldr    Index,u      get table Index
: leax   table,pcr    get table base address
: add    Index,u      multiply Index by two
: ldd    d,x          get table entry
:
:
: fdb    addr1-table  table entries
: fdb    addr2-table
: fdb    addr3-table
: fdb    addr4-table
:
:

```

If the address of the beginning of the module is used as a base point, an instruction such as the following:

```
leax start,pcr
```

would be inserted between the two "ldd" instructions and the "table" names would be dropped from the table entries, or changed to "start".

In the case of complex symbolic expressions involving several labels, the assumed bases of each of the labels in the expression must be checked to assure that the expression is meaningful when the program is position-independent in code and data.

For example, in the following program fragment:

```

:
:
: ubrel  rmb 1        U-relative label
: ubrel1 rmb 1
: ubrel2 rmb 1
:
:
: pcrel  leax (pcrel-ubrel),pcr    pc-relative label
: urel   leax (urel-ubrel1),u      U-relative label
:
:

```


the values of symbolic expressions of the form "(pcrel-label)" and "(urel-label)" are meaningless with either "PCP" or "UP" suffix, since the offset between the program module load point and the start of the data space (represented by the U register) is not constant.

Expressions must normally be built using symbols based upon the same register. However, expressions or sub-expressions of the following formats:

```
lulabel1-label2l
(urel-pcrel)
```

are meaningful since they specify constant values, assuming both symbols are based upon the same register, as in the example above.

There are several differences between the FLEX assemblers and OS/9 assemblers which must be understood to read the example OS/9 programs provided later in this article. The first difference is found in the library call pseudo-opcode. FLEX assemblers require "lib", whereas OS/9 assemblers require "use". The "opt" pseudo-opcode operands are also different among the assemblers. OS/9 assemblers use "." rather than "m" to designate the data counter, which is analogous to the program counters used by FLEX and OS/9 assemblers.

FLEX PROGRAM REQUIREMENTS

FLEX places very few requirements on application programs written to execute under its control. There are two areas into which FLEX programs are normally loaded, as follows:

```
$0000-memend application program area
$cl00-$c6ff utility program area
```

where "memend" represents a value maintained by FLEX which specifies the end of the user address space. The value of "memend" may be changed by FLEX system programs and by device drivers stealing space from the user area in order to become memory-resident.

The FLEX representation of an object program code extent on disk is as follows:

```
$02 Indicator of code extent
high address high byte of extent address
low address low byte of extent address
count number of bytes in extent
data contents of extent
```

and the FLEX representation of the transfer address of an object program is as follows:

```
$16 Indicator of transfer address
high address high byte of transfer address
low address low byte of transfer address
```

where a given object file may be composed of any number of extents and transfer addresses. Each extent is loaded independently of all others, providing the facility of discontinuous program code loading. The last-found transfer address is used as the initial program execution address. A transfer address is normally required in order to execute a program.

FLEX has no protection against an application or system program being loaded over itself or over other memory-resident programs or device drivers. It is the user's responsibility to ensure that the program address spaces do not conflict. Since FLEX is inherently single-user, this does not usually cause problems. Programs such as BASIC and assemblers use "memend" to determine the end of the application program area in order to utilize the maximum amount of memory available.

FLEX device drivers (and certain other system programs) must possess position-independency, but FLEX does not take any special actions for position-independent application programs. In fact, since position-independent OS/9 programs are generally longer, slower, and harder to write than those that are not, most FLEX application programs are not position-independent.

Although FLEX supports only a limited number of devices, system and application programs have access to all of the 64K bytes of the address space, and are free to implement any internal devices, as required. FLEX will not recognize these devices unless the programs are written as device drivers. In this case, the drivers are recognized as either a output or input device or disk device; however, disk devices are designated as drives numbered zero thru three, restricting the number of disk drives to four.

OS/9 IMPLEMENTATION OF FLEX AND SBUG ENTRY POINTS

This section discusses the implementation of FLEX assembly language programs in OS/9 assembler language, primarily in terms of the conversion of the FLEX and SBUG entry points. In many cases, the suggested conversions are approximate, and problems are usually noted as such.

All the conversions are symbolic and indicative of the logic conversions required, not necessarily indicative of the specific conversions necessitated by a given situation in a specific program. Undocumented side-effects of FLEX entry points are not implemented in the suggested conversions. Unused registers are maintained.

The use of FLEX and SBUG entry points not included in the list below must be carefully investigated; most are not convertible to OS/9 and logic using them must be deleted or rewritten.

Note that all OS/9 console I/O may be redirected from the command line, whereas FLEX has entry points and switches which can override console I/O redirection. This may effect logic in many programs being converted which use FLEX and SBUG entry points such as the following:

```
inch/inch2
outch/outch2
getchr
putchr
finch
foutch
finch2/finch2
```

those programs using other FLEX entry points with I/O to the console, and those programs which use the FLEX switches "inswt" and "outswt" to control redirection of console I/O.

The problems involved in console I/O redirection under OS/9 and the differences between the handling of the command lines and file names of FLEX and OS/9 may also cause logic changes in programs using the following FLEX entry points to scan the command line:

```
inbuff
nxtch
getfll
setext
```

The conversions described below may be used in several different manners.

In short programs, the simplest technique may, in many cases, be to substitute the equivalent code directly on each occurrence of the calls. This has the advantage that it may be easier to understand and to accomplish. Complicating factors in this method may be secondary entry points (in which one conversion routine calls another), changed code sequence length (requiring the substitution of long branches for short branches), and loss of structure and modularity of the program code.

In longer programs, one copy of each of the OS/9 equivalent code groups used in each program could be included and invoked with "lib" or "lbr", replacing the "jmp" or "jsr" in the original FLEX program code. Indirect jumps and subroutine calls to the FLEX and SBUG entry points must be changed to the corresponding relative branches.

Still another technique would be to use a complete emulator to run FLEX programs effectively as-is under OS/9. This would not accomplish the conversion of FLEX programs to OS/9 programs, but would allow the use of FLEX programs under OS/9, with limitations.

The technique suggested in the descriptions below involves the use of a conversion subroutine library to provide facilities similar to those provided by the original FLEX program code.

SUGGESTED CONVERSIONS

The following entry points return to FLEX or SBUG. The OS/9 equivalent returns to OS/9 with a zero return code, which may be modified to indicate various error or logical conditions.

Entry Point	Parameters	Conversion	Comments
colds		warms clrb	
warms		os9 F\$Exit	
center			
monitor			
monitcr			
nextcmd			

The following entry points wait for, input, and echo a character from the FLEX or SBUG console input device. The OS/9 equivalent performs the same action; however, standard input redirection may cause problems in reading from the desired OS/9 console input device. Also, the OS/9 console drivers may mask character parity, ignore certain character codes, and otherwise perform differently than the FLEX console drivers; many of these actions may be changed by modifying the OS/9 device descriptor for the console.

Entry Point	Parameters	Conversion	Comments
inch (FLEX) returns	getchr	psht b,x,y,u	
inch2	a:char	leas -\$01,s	
inchc		tfr s,x	
finchc		ldy #00001	
		clra	
		os9 I\$Read	
		lda s,x	
		puls b,x,y,u,pc	
getchr	returns a:char	getchr psht b,x,y,u	
		leas -\$01,s	
		tfr s,x	
		ldy #00001	
		clra	
		os9 I\$Read.n	
		lda s,x	
		puls b,x,y,u,pc	

The following entry points wait for and input a character from the FLEX or SBUG console input device. The OS/9 equivalent performs the same action; however, standard input redirection may cause problems in reading from the desired OS/9 console input device. Also, the OS/9 console drivers may mask character parity, ignore certain character codes, and otherwise perform differently than the FLEX console drivers; many of these actions may be changed by modifying the OS/9 device descriptor for the console.

Entry Point	Parameters	Conversion	Comments
inch (SBUG) returns	finch	psht b,x,y,u	
finch	a:char	leas -\$20,s	
		tfr s,x	
		clra	
		clrb	
		os9 I\$GetStt	
		clr pd.eho-\$20,x	
		os9 I\$SetStt	
		leas \$20,s	
		ldy #00001	
		clra	
		os9 I\$Read	
		tfr s,x	
		clra	
		clrb	
		inc pd.eho-\$20,x	
		os9 I\$SetStt	
		leas \$20,s	
		puls b,x,y,u,pc	

The following entry points check for input ready on the FLEX or SBUG console input device. The OS/9 equivalent performs the same action; however, standard input redirection may cause problems in reading from the desired OS/9 console input device.

Entry Point	Parameters	Conversion	Comments
stat	returns	stat psht b,x,y,u	

```

tcheck      z=ready      ldd      $0001
lcheck      os9           l$GetStt
            andb          #02
            comb
            puls          d,x,y,u,pc

```

The following entry points output a character to the FLEX or SBUG console output device. The OS/9 equivalent performs the same action; however, standard output redirection may cause problems in reading from the desired OS/9 console output device. Also, the OS/9 console drivers may mask character parity, add line feeds after carriage returns, and otherwise perform differently than the FLEX console drivers; many of these actions may be changed by modifying the OS/9 device descriptor for the console.

Entry Point	Parameters	Conversion	Comments
outch outch2 touch	a=char	outch pshs d,x,y,u tfr s,x ldy \$0001 lda \$01 os9 l\$wrt puls d,x,y,u,pc	
putchr	a=char	putchr pshs d,x,y,u tfr s,x ldy \$0001 lda \$01 os9 l\$wrt puls d,x,y,u,pc	

The following entry points output a string of characters to the FLEX or SBUG console output device. The OS/9 equivalent performs the same action; however, standard output redirection may cause problems in reading from the desired OS/9 console output device. Also, the OS/9 console drivers may mask character parity, add line feeds after carriage returns, and otherwise perform differently than the FLEX console drivers; many of these actions may be changed by modifying the OS/9 device descriptor for the console.

Entry Point	Parameters	Conversion	Comments
pstring	x=>string	pstring bsr pcrif bsr pdata rts	
pdata	x=>string	pdata lds ,x+ cmpa \$04 beq pdatex lbrs outch bra pdata rts	
pcrif	pcrif	pshs a lda \$0d lbrs putchr puls a,pc	

The following entry point waits for inputs, and echoes a line from the FLEX console input device. The OS/9 equivalent performs the same action; however, standard input redirection may cause problems in reading from the desired OS/9 console input device. Also, the OS/9 console drivers may mask character parity, ignore certain character codes, and otherwise perform differently than the FLEX console drivers; many of these actions may be changed by modifying the OS/9 device descriptor for the console.

The buffer used by the routine must be defined as at least 128 bytes in length and should be initialized to the contents of the command line, as shown below, before any of the registers passed to the module have been modified.

Entry Point	Parameters	Conversion	Comments
Inbuff	returns Inbuff	Inbuff pshs d,x,y,u leax Inbuff,u stx cbufl,u ldy \$07f clr os9 puls d,x,y,u,pc	
Initial	Initial	pshs d,x,y leay Inbuff,u initl lds ,x+ sta ,y+ cmpa \$0d bne initl puls d,x,y,pc	

The following entry point classifies the character represented by the A register. If the character is alphabetic or numeric, the routine returns with the carry flag cleared. Otherwise, it returns with the carry flag set and the character in the last terminator location.

Entry Point	Parameters	Conversion	Comments
class	a=char returns cc=class	class cmpa \$030 bcs classn cmpa \$039 bls classe cmpa \$041 bcs classn cmpa \$05a bls classe cmpa \$061 bcs classn	

```

cmpa      $07a
bhl       classn
classn    andcc  $01a
rts
classn    orcc   $01
sta      istrm,u
rts

```

The following entry point returns the next character from the input buffer (constructed by the "Inbuff" entry point) in the A register. It skips multiple spaces and also classifies the character, but will not scan beyond the end of the logical or physical line.

Entry Point	Parameters	Conversion	Comments
nxtch	returns a=char cc=class	nxtch pshs b,x,y,u lda curchr,u sta prvchr,u ldx Inbuff,u nxtchl lda ,x+ sta curchr,u cmpa ttyeof,u beq nxtch2 cmpa \$0d beq nxtch2 ste Inbuff,u cmpa \$020 bne nxtch2 cmpa ,x beq nxtchl nxtch2 lbrs class puls b,x,y,u,pc	

The following entry point converts the 16-bit unsigned integer pointed to by the X register to decimal and outputs it to the FLEX console output device. If the B register is non-zero on entry, the routine will output leading spaces; otherwise, it will suppress leading spaces and zeroes.

Entry Point	Parameters	Conversion	Comments
outdec	x=>number	outdec pshs d,x,y,u ldd \$03004 pshs d ldd ,x leax outdes,pcr ldy \$0000 outdex cmpd ,x bhs outdeg outdex pshs d ldd \$02,s cmpy \$0000 bne outdeg fsta outdel bne outdel fsta outdel beq outden lda \$020 bra outdex outdel leay \$01,y outdeg lbrs putchr outden lda \$030 sta ,s puls d dec \$01,s beq outdex leax \$02,x bra outdex outdeg lnc ,s subd ,x bra outdex outdex puls d tfr b,s ora \$030 lbrs putchr puls d,x,y,u,pc outdes fdb 10000 fdb 1000 fdb 100 fdb 10	

The following entry points convert the 16 bit or 8 bit number pointed to by the X register to hexadecimal and outputs it to the FLEX console output device.

Entry Point	Parameters	Conversion	Comments
outadr	x=>number	outadr bsr outhex leax \$01,x bsr outhex rts	
outhex	x=>number	outhex lda ,x bsr outhx1 lda ,x bra outhx2 outhx1 lsr lsr lsr lsr outhx2 anda \$0f adda \$030 cmpa \$039 bls outhx3	

The Best 30 Pin Floppy Controller You Can Buy!

The ELEKTRA Super Floppy Controller

Run:

5"

8"

Single Sided

Double Sided

Single Density

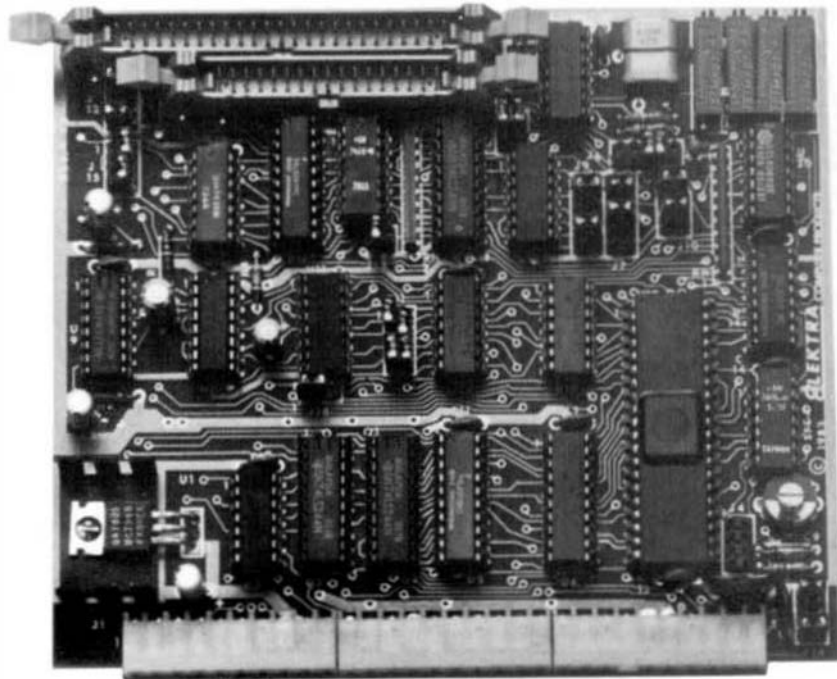
Double Density

5" and 8"

Mixed Systems

1 MHz

2 MHz



(8" Double Density must run at 2 MHz)

Phone

AAA Chicago Computer Center

Technical consultation available most weekdays from
4 p.m. to 6 p.m. C.S.T.

(312) 459-0450

120 Chestnut Lane, Wheeling, IL 60090

See our catalog and ordering information on the next page to your right.

HELIX

64K 6808 Computer	\$2395.00	84K 6808 Computer	\$2495.00
256K 6809 Computer	2895.00	256K 6809 Computer	2995.00
Other systems available			
20 Megabyte 5" add on Winchester System	\$2595.00		
64K CMOS Static memory board with battery backup	385.00		
DMA 5" and 8" Floppy Controller with built in Winchester controller I/O	585.00		
DMA 5" and 8" Floppy Controller	\$495.00	6809 CPU Board	495.00
6809 board for SS-50	595.00	CPU-6809	350.00

Need FLEX, UniFLEX, OS-9 Level I, or OS-9 Level II? We have a system for you!

ELEKTRA COMPUTER CABINET THE LARGEST SS-50 COMPUTER CABINET AVAILABLE! Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for D type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and Null/Abort), and two cutouts for 5-1/4" disk drives.

Filter Plate for 5-1/4" drive opening. \$10.00 Fan Filter \$10.00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ 16v. 3 primary inputs for high, rated, and heavy loading. 220v Version. \$200.00 110v Version. \$175.00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives \$50.00 Heavy duty version for 1 Winchester drive and 1 floppy drive. \$75.00

AUXILIARY POWER SUPPLY to power cond Winchester drive \$125.00

ENGINEER'S "FUN BOX" BY ELEKTRA Computer cabinet with high quality 10 amp power supply, EMI filter, fan. Large enough to hold the standard size SS-50 type motherboard. \$225.00

ELEKTRA UNIVERSAL SS-50/SS-50C MOTHERBOARD Heavyweight 0.125" thick, 18" long by 9" wide, 11 memory (50 pin) slots, 8 I/O (30 pin) slots. Complete address decoding and selection, as well as extended address capability for I/O slots. Choice of 4, 8, or 16 addresses per I/O slot, 1" spacing between all memory and I/O slots. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines, above device circuitry permitting 1 MHz 30 pin data controllers to run with 2MHz 50 pin CPU boards.

Mounting hardware \$5.00 Bareboard w/documentation \$80.00 Kit w/gold connectors \$320.00 Assembled w/gold connectors \$380.00 Kit w/in connectors \$240.00 Assembled w/in connectors \$300.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. \$850.00

ELEKTRA CPU 6809 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2716 Eproms, 1K scratchpad, M68040 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. Versions of OS-9 level I are available.

Bareboard \$30.00 Kit \$225.00 Assembled \$275.00 Optional Baud Rate Generator \$25.00

ELEKTRA DPS/DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per port. RTS, CTS, DTR, DCD, IRO, FIRQ, NMI, and baud rate can be appropriately implemented for each port.

Bareboard \$20.00 Kit \$80.00 Assembled \$80.00 Cable with mounting hardware (two needed per board) Each \$25.00 Cable Each 20.00 Mounting hardware per cable \$5.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request line for each port may be individually jumpered to either the IRO or FIRQ/NMI bus line.

Bareboard \$20.00 Kit \$80.00 Assembled \$80.00 Cable with mounting hardware (two needed per board) Each \$25.00 Cable Each 20.00 Mounting hardware per cable \$5.00

ELEKTRA 64K STATIC RAM/ROM MEMORY BOARDS with gold connectors (for availability) Assembled and tested. With 56K RAM \$269.00 With 64K RAM \$299.00

ELEKTRA UNIVERSAL SUPER FLOPPY CONTROLLER THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! Controls up to four 5-1/4" drives and four 8" drives for a total of eight system drives. (FLEX system limit is four drives.) Single density or double density, 1MHz or 2MHz, 6800 or 6809. (Double density 8" must be at 2MHz, all other combinations of performance are possible.) Analog phase locked loop data separators with separate adjustments for 5" and 8" drives. Analog write precompensation circuit with separate adjustments for 5" and 8" drives. Designed to meet the data hold requirements of Western Digital floppy controller IC. Assembled and tested. \$275.00

Disk with drivers and formatting utilities (Specify 6800/9, FLEX/OS-9) 30.00

ELEKTRA WINCHESTER SYSTEMS THE BEST WINCHESTER SYSTEMS THAT YOU CAN BUY! Has automatic error detection and CORRECTION of up to 11 bit burst errors. SS-50 bus, extended addressing capabilities, DMA, on board sector buffer, drivers included for 6808 FLEX or OS-9. Specify whose version of FLEX that you are using. Drivers for FLEX2 (6800) are available for an additional \$100.00. Price includes host interface, controller, drive(s), and cables.

12 Megabyte single drive sys. \$2295.00 24 Megabyte dual drive sys. \$359.00 19 Megabyte single drive sys. \$2995.00 38 Megabyte dual drive sys. \$4695.00 (19 Megabyte drives are the largest that can be supported FLEX)

ELEKTRA HD-5 Cabinet for dual 5 1/4" floppy drives with power supply, line cord, fuse power switch, and power cables to drives. \$150.00

ELEKTRA HD-5W As above but with EMI filter, fan, and heavy duty power supply. Powers 1 floppy and 1 Winchester. 199.00

5" ribbon cable for dual outboard 5-1/4" disk drive 40.00 2" ribbon cable for dual inboard 5-1/4" disk drives 35.00 Custom cables available Phone

ELEKTRA HD-8 Dual drive cabinet, EMI filter, fan, power supply, and power supply cables for 8" drives. 350.00 6 ribbon cable for dual 8" disk drive 45.00

ELEKTRA 30 PIN PROTOTYPING BOARD 20.00

ELEKTRA 30 PIN PROTOTYPING BOARD 40.00

GGLO 10 PIN CONNECTORS (Specify male with square pins or female) 1.50

TIN 10 PIN CONNECTORS (Specify male with square pins or female) .50

ELEKTRA is a trademark of AAA Chicago Computer Center. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. HELIX is a trademark of Hazelwood Computer Systems. OS-9 and BASIC09 are trademarks of Microware Systems Corp.

Dealer for ELEKTRA, HELIX, SSB, SWTPC, Microware Systems Corp., and Technical Systems Consultants, Inc.

AAA CHICAGO COMPUTER CENTER 120 CHESTNUT LANE • WHEELING, IL 60090 13121 458-0450

Technical consultation available 4 PM to 6 PM m. at weekdays. Closed evenings and weekends.

TERMS Minimum order \$20.00. Shipping and handling estimates within the Continental U.S. add 3% (MINIMUM \$2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM \$10.00). Foreign orders must be prepaid in U.S. dollars. Checks must be drawn on a U.S. bank. Please allow 4-6 weeks for items to be shipped air freight collect. Please phone between 4 PM and 6 PM weekday if questions are regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

ELEKTRA SOFTWARE (All of our software is copyrighted and all rights are reserved. Source is either supplied or optionally available at extra cost so that the purchaser can modify our programs for his own use. Licensing, however, is required for commercial resale.)

SUPER MODEM PROGRAM Single character commands. No interrupts required. Transmit manually or in semi-duplex. Files (text) of any length to distant computer. Receive and save disk files (text) on local disk system. X-on/X-off supported. Tested for full duplex at speeds up to 9600 baud. Half duplex option. Echo option. Resolves CR with CR/LF (user option). Slow disk file transmit option.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8" Instruction Manual and disk with both source and object code \$75.00

STANDARD MODEM PROGRAM Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Specify 6800 or 6809. Manual with instructions, source listing, and flow chart. 10.00

OS-9 Configurable Modem Program (Sorry, source is not available) 100.00

ORDER — WRITE UP COMPUTER PROGRAM Screen oriented write up form with cursor editing, disk save and load, printer command using easily available universal print-out forms. Phone for more details. Available for 6809 FLEX. \$100.00

ALL IN ONE Editor — Text Processor — Mailing Labels — Mailing Lists — Multiple Form Letters. Use any CRT terminal and printer — Best Package For The Money Anywhere! \$75.00

Specify 6800 or 6809, SSB or FLEX™, 5" or 8" Printed source listing is available for an additional \$35.00

All-in-One, Write'n spell, and Spell'n Fix package 250.00

Software by Technical Systems Consultants, Inc.

Software	Adapt. Guide	FLEX(TM)				UniFLEX(TM)			
		Source (List)	Source (Disk)	Man. Only	Object w/Man.	Man. Only	Man. Only	Object w/Man.	Object w/Man.
Gen. DOS w/Editor & ASMB	25	—	—	25	250	—	—	—	—
SWTPC DOS w/Editor & ASMB	—	—	—	25	150	40	100	560	—
Advanced Programmers Guide	—	—	—	25	—	—	—	—	—
Editor	100	250	25	50	—	—	—	—	—
Assembler	150	250	25	50	—	—	—	—	—
Debugger	17.5	250	25	75	—	—	—	—	—
Extended Basic	—	—	—	25	100	20	50	200	—
Basic Precompiler	—	—	—	25	50	10	25	150	—
Sort/Merge	—	—	—	25	75	20	35	150	—
Utilities	—	Inc	—	25	75	10	25	150	—
Diagnostics	—	—	—	25	75	—	—	—	—
Text Processor	150	250	25	75	20	35	150	—	—
68000 X-ASMB on 6809	—	—	—	25	250	20	35	300	—
Pascal	—	—	—	50	200	25	50	300	—
Rel. SMB/Linking Loader	—	—	—	25	150	20	35	175	—
6800 X-ASMB on 6809	—	—	—	100	—	—	—	—	—
Cobol	—	—	—	—	30	75	750	—	—
Fortran 77	—	—	—	—	35	85	450	—	—

Software by Microware Systems Corp.

Software	Run-Time Package	Update	Source	Manual	Object Only w/Man.
OS-9™ Level One Operating System	75.00	400.00	—	40.00	200.00
OS-9™ Level Two Operating System	75.00	N/A	—	40.00	500.00
BAS-9™	100.00	75.00	N/A	25.00	200.00
OS-9™ Macro Text Editor	—	—	300.00	15.00	125.00
OS-9™ Interactive Assembler	—	—	300.00	10.00	125.00
OS-9™ Interactive Debugger (I/O version)	—	—	100.00	10.00	50.00
CCS Cobol Compiler	400.00	50.00	N/A	80.00	900.00
Pascal Compiler	100.00	100.00	N/A	40.00	400.00
C Compiler	100.00	100.00	N/A	40.00	400.00
Microware yearly support service (\$ 00 for OS-9 Level 2)	—	—	—	—	75.00

Special Software

2K 6809 MICROBUG 30.00 4K 6809 HUMBUG 75.00 2K 6800 HUMBUG 40.00 4K 6800 HUMBUG 65.00

Other HUMBUG versions including video versions are available. Spell'n Fix by Peter Stark 178.58 Write'n Spell by Peter Stark 75.11

All-in-One, Spell'n Fix, and Write'n Spell packages 250.00

Dynalite Disassembler 60.00

SUPER SLEUTH Disassembler System (\$100.00 for OS-9 version) 99.00

SO/DISK DRIVES

30 day guarantee 1 head 2 heads 2 heads 1 head 2 heads

5-1/4" 40 tracks 225.00 300.00 300.00 250.00 325.00

5-1/4" 80 tracks 300.00 375.00 375.00 325.00 400.00

MPI or C DC Service Manual (Specify 40 or 80 track) 25.00

8" 77 tracks OS/D 0ume DT-6 \$550.00 Remex (Special) 350.00

SPECIAL

* U.S. Robotics 300/1200 baud auto dialer to answer modem 499.00

* Same as above but without self test and diagnostics 399.00

* U.S. Robotics 1200 baud direct connect auto answer modem 349.00

* Standard 1420 CRT terminal (new) 375.00

* SSB BFD Floppy Disk Controller (Version 3) Run FLEX or SSB DOS 100.00

* SWTPC 4K Memory \$15.00 MP-B \$40.00 MP-A with MP-C 35.00

* High speed tape reader 50.00 AC-30 \$40.00 MP-S 60.00

* 300 Baud Acoustic modem 129.00

* 10 ELEKTRA DD 8" Disks 35.00 10 5" DD Disks in hard box 25.00

* T1 810 Printer w/lower case and full vertical forms control 1200.00

* Micr time II Calendar and Clock Board (Assembled) 80.00

GIMIX CLEARANCE SALE

OUR LIST PRICE OUR LIST PRICE

#5 6809 Plus CPU Bd. 578.05 475.00 8800 CPU board 224.03 100.00

Cable (Ser or Par I/O) 24.95 20.00 Motherboard 200.00

Double disk reg. card 68.22 50.00 8 Port Serial I/O Bd. 318.46 250.00

32K memory board 175.00 #28 control w/GMX Flex 328.28 270.00

64K memory board 478.67 450.00 56K memory board 425.00

80 X 24 Video Boards 398.76 250.00 Single prt ser. 1 cable 113.36 90.00

84 X 16 Video Boards 198.71 108.00 Dual prt ser. 2 cables 138.32 110.00

18R Mem Bds w/cmri reg. 45.00 4K PPD PROM Bd. and burner 100.00

90L422 RAM chips (2 needed for GIMIX OAT) each 20.00

SWTPC

DC-4 5-1/4" Disk Controller 230.00 SWTPC 6809 FLEX™ Disk & Man. 35.00

MP-52 Dual Port Serial 120.00 MP-L2 Dual Port Parallel 120.00

MP-N Calculator Board (kit) 54.95 MP-N (assembled) 92.00

MP-R 2716 Eprom Programmer 11a 50 MP-09 2MHz 6809 CPU Board 285.00

Smoke Signal Broadcasting

DCB-4A Double Density Controller Board for 5" and 8" with DOS 549.00

SSB DOS (Specify 6800 or 6809, 8FD or DCB-4A, 5" or 8") 75.00

SE92/SA92-5 (6809 Edit/Asm for DOS) 89.95

SSB Monitor (Specify 6800/6809, 8800/6809/67E8) 75.00

SSB version of FLEX™ (without Editor and Assembler) 150.00

LMB-1A Motherboard 399.00

SCB-69 6809 CPU Board 399.00

PAR-1 Dual Port Parallel Board 85.00

SER-2 Dual Port Serial Board with 2 Cables 129.00

Chief 90 64K Computer System 2195.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating out kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drives, and software along with I/O terminals to form working computer systems.


```

      adda    #007
outhx3 lbra  putchr

```

The following entry points convert a hexadecimal or decimal number in the input line buffer and return the result in the X register. The carry bit is cleared on exit if a valid number was found.

Entry Point	Parameters	Conversion	Comments
gethex	returns x=number cc=valid	gethex pshs d,y,u ldx #00000 gethx1 lbrs natch bcs gethx6 cmpa #55f b1s gethx2 suba #520 gethx2 suba #547 bpl gethx5 adda #506 bpl gethx3 adda #007 bpl gethx5 gethx3 adda #50a bml gethx5 exg d,x asl b rol a as1 b rol a asl b rol a asl b rol a neg leax a,x bra gethx1 gethx5 lbrs natch bcc gethx5 puls -1,y,u,pc gethx5 andcc #51a puls d,y,u,pc	

Indec	returns x=number cc=valid	Indec pshs d,y,u ldx #00000 Indec1 lbrs natch bcs Indec3 cmpa #539 bhl Indec2 pshs x as1 b rol a asl b rol a asl b rol a adda .5 adda .5+ exg d,x andx #50f leax a,x bra Indec1 Indec2 lbrs natch bcc Indec2 puls d,y,u,pc Indec3 andcc #51a puls d,y,u,pc	
-------	---------------------------------	---	--

The following entry point adds the contents of the B register to the contents of the X register and returns the result in the X register. It was included in FLEX 9 for compatibility with FLEX 2 on the 6800.

Entry Point	Parameters	Conversion	Comments
addbx	x+b=x	addbx abx rts	

The following entry point formats an error message and outputs it to the FLEX console output device. The error number is found in the FCB error field to be the X register.

Entry Point	Parameters	Conversion	Comments
rpterr	x=>FCB	rpterr pshs d,x,y,u lda #502 ldh \$01,x os9 FSPerr puls d,x,y,u,pc	

The following entry points transfer a FLEX file name from the line buffer to an FCB to which the X register points and provide a default file name suffix. If the format of the file name is not valid, the carry flag will be set on exit from the routine and the error field in the FCB will be set to 21 (Illegal File Specification). However, since the formats and lengths of the file names are quite different between the two systems, there can be no direct equivalent under OS/9.

The conversion suggested below combines the external functions of the two FLEX entry points into one routine which locates an OS/9 path and file name in the line buffer, points the Y register to the beginning of the file name, and sets the A register to the length of the file name. If the format of the file name is not valid, the carry flag will be set, the FCB error field will be set to 21, and the A register will be set to zero. The other changes caused by these differences must be handled separately.

Entry Point	Parameters	Conversion	Comments
-------------	------------	------------	----------

getf11 setext	x=>FCB	getf11 pshs b,x,u ldx cbufpt,u clra getf11 tlr x,y ldh .x+ cmpb #520 beq getf11 cmpb #52a beq getf12 cmpb #52f beq getf12 cmpb #541 blo getf13 cepb #55a b1s getf12 cmpb #55f beq getf12 cmpb #561 blo getf13 cmpb #57a bhl getf13 getf12 lncx .x+ ldh #52a cmpb getf12 cmpb #52f beq getf12 cmpb #541 blo getf13 cmpb #55a b1s getf12 cmpb #55f beq getf12 cmpb #561 blo getf13 cepb #57a b1s getf12 getf13 cmpb #50d beq getf14 cmpb #521 beq getf14 cmph #52a beq getf14 bne getf15 getf14 leax -501,x getf15 stx cbufpt,u tsa beq getf17 getf16 andcc #51a puls b,x,u,pc getf17 ldx #515 ldh \$01,x stb \$01,x orcc #501 puls b,x,u,pc	
------------------	--------	---	--

The following entry point loads a binary file the name of which is contained in the system FCB lsrscbl. The nearest OS/9 equivalent of the FLEX entry point requires that the X register point to the name of the file, advances the X register past the name, and sets the carry flag and B register in case of error.

Entry Point	Parameters	Conversion	Comments
load		load os9 F\$Load bcc loadx os9 F\$Perr loadx rts	

The following entry point passes the line buffer to FLEX as a command line. FLEX processes the contents of the line buffer and returns the last FMS error code in the B register. The primary differences between the FLEX entry point and the nearest OS/9 equivalent lie in the command line formats, the error code interpretations, and the new process id returned in the A register.

Entry Point	Parameters	Conversion	Comments
docmd	inbuff	docmd pshs x,y,u leax docmd.pcr ldy #5007f leau inbuff,u clre clrb os9 F\$fork bcs docmdx os9 F\$wait docmdx puls x,y,u,pc docmdx fcs "Shell"	

The following entry points perform all disk-related operations for the remainder of FLEX and for application programs executing under control of FLEX. As noted earlier, all communications to the File Management System (FMS) are thru the File Control Block (FCB) and all communications from the FMS are thru the FCB and the zero flag.

There are very few OS/9 system request equivalents for the FLEX FMS function codes. There is no direct equivalent for the FCB. Since many of the FMS functions use or deposit data from or into the FCB, the FCB is assumed maintained in the suggested conversions listed below. However, the OS/9 path name is potentially much longer than the FLEX file name, must be maintained separately from the FCB, and its address must be passed in the Y register.

Any FMS functions dealing with the FLEX SRR, random files and records, the directory, or full sectors cannot be directly translated to

OS/9. Any references to these functions are highly system-dependent and must be carefully investigated and modified or eliminated. Often, OS/9 provides a direct means to accomplish what in FLEX is fairly difficult or obscure. An example of this is that OS/9 allows the direct reading of a directory as if it were a file, whereas FLEX has separate FMS entry points for reading the directory.

The conversions below assume that the FMS functions will be separate. An alternative organization would be to provide one entry point and have logic there to separate the functions by the FCB function code; however, in the majority of cases, definite FMS functions are called, making a single entry point superfluous.

Thus, this area of conversion from FLEX to OS/9 will probably require more effort in a given program than all the remainder of the areas of conversion.

Entry Point	Parameters	Conversion	Comments
fmsini		fmsini rts	Initialize FMS
fmscls		fmscls rts	close all files
fms	fcn=00 get/put a character x=>FCB	fms00 pshs d,x,y,u clr \$01,x ldd \$02,x tfr \$x,x ldy \$00001 cmpa \$02 tfr b,e bne fms00r os9 i\$write bra fms00t fms00r os9 i\$Read fms00r orcc \$04 bcc fms00x ldr \$02,s stb \$01,x fms00x puls d,x,y,u	get/put byte status and path
fms	fcn=01 y=>name x=>FCB	fms01 pshs d,x,y,u clr \$01,x lda x sta \$02,x lda \$01 exg x,y os9 i\$Open sta \$03,y orcc \$04 hnc fms01x clr \$03,y stb \$01,y fms01x puls d,x,y,u	open input \$05 for binary
fms	fcn=02 y=>name x=>FCB	fms02 pshs d,x,y,u clr \$01,x lda x sta \$02,x ldd \$021b exg x,y os9 i\$Create sta \$03,y orcc \$04 bcc fms02x clr \$03,y stb \$01,y fms02x puls d,x,y,u	open output \$063 for binary
fms	fcn=03 y=>name x=>FCB	fms03 pshs d,x,y,u clr \$01,x lda x sta \$02,x lda \$03 exg x,y os9 i\$Open sta \$03,y orcc \$04 bcc fms03x clr \$03,y stb \$01,y fms03x puls d,x,y,u	open update \$07 for binary
fms	fcn=04 x=>FCB	fms04 pshs d,x,y,u clr \$01,x clr \$02,x lda \$03,x clr \$03,x os9 i\$Close orcc \$04 bcc fms04x stb \$01,x fms04x puls d,x,y,u	close path
fms	fcn=05 x=>FCB	fms05 pshs d,x,y,u clr \$01,x lda \$01,x tfr x,y ldx \$0000d tfr x,u os9 i\$Seek orcc \$04 bcc fms05x stb \$01,y fms05x puls d,x,y,u	rewind path
fms	fcn=06 x=>FCB	fms06 rts	open directory no OS/9 equivalent
fms	fcn=07	fms07 rts	get info record

	x=>FCB	*	no OS/9 equivalent
fms	fcn=08 x=>FCB	fms08 *	rts put info record no OS/9 equivalent
fms	fcn=09 x=>FCB	fms09 * * *	rts read single sector no OS/9 equivalent (could open device for physical sector I/O and use i\$Seek)
fms	fcn=0a x=>FCB	fms0a * * *	rts write single sector no OS/9 equivalent (could open device for physical sector I/O and use i\$Seek)
fms	fcn=0b x=>FCB	fms0b *	rts extend directory no OS/9 equivalent
fms	fcn=0c y=>name x=>FCB	fms0c pshs clr exg os9 orcc bcc stb fms0cx puls	d,x,y,u delete file \$01,x x,y i\$Delete i\$DeleteX for binary \$04 fms0cx \$01,y d,x,y,u
fms	fcn=0d x=>FCB	fms0d *	rts random file no OS/9 equivalent
fms	fcn=0f x=>FCB	fms0f *	rts next next sector no OS/9 equivalent
fms	fcn=10 x=>FCB	fms10 *	rts open sys info rec no OS/9 equivalent
fms	fcn=11 x=>FCB	fms11 * *	rts get random byte no OS/9 equivalent (could use i\$GetStt and i\$Seek)
fms	fcn=12 x=>FCB	fms12 * *	rts put random byte no OS/9 equivalent (could use i\$GetStt and i\$Seek)
fms	fcn=13 y=>name x=>FCB	fms13 pshs clr lda sta lda exg os9 sta bcc clr stb puls fms13a ldb os9 orcc bcc stb puls fms13x cmpx bne cpu beq fms13a lequ cpu bne leax fms13t os9 orcc bcc stb fms13x puls	d,x,y,u open extend \$01,x \$02 \$02,x \$02 x,y i\$Open \$03,y fms13a \$03,y \$01,y d,x,y,u \$03,s12 i\$GetStt \$04 fms13x \$01,y d,x,y,u \$0000 fms13a \$0000 fms13t \$01,x i\$Seek \$04 fms13x \$01,y d,x,y,u
fms	fcn=14 x=>FCB	fms14 *	rts find next drive no OS/9 equivalent
fms	fcn=15 x=>FCB	fms15 * *	rts position to record n no OS/9 equivalent (could use i\$Seek to byte position)
fms	fcn=16 x=>FCB	fms16 * *	rts back up one record no OS/9 equivalent (could use i\$Seek to byte position)

OS/9 IMPLEMENTATION OF FLEX STORAGE LOCATIONS

This section discusses the implementation of the FLEX storage locations under OS/9, assuming the conversions already described are used. As before, some of the suggested storage location conversions are approximate, because of the different orientation OS/9 takes in many areas from FLEX. The use of locations not on this list must be investigated thoroughly and modified or dropped.

name	description	comments
stack	FLEX stack	The FLEX stack area is normally used by programs requiring only a small stack. Explicit reference to the stack may be used to reset it to a known value. If necessary, the initial value of the stack could be saved in order to implement such stack-resetting logic.

inbuff command line buffer
The command line buffer is used for communication between FLEX and user programs. It must be at least 128 bytes in length and should be initialized to the contents of the parameter area.

sysfcb system file control block
The system FCB is used by many programs as a temporary additional FCB. A 320 byte area named "sysfcb" may be substituted for the system FCB.

bspcbr ttyset backspace
delchr ttyset delete
eofchr ttyset end of line
depth ttyset depth count
width ttyset width count
nulls ttyset null count
tabchr ttyset tab
bspcbr ttyset backspace echo
ejct ttyset eject count
pause ttyset pause control
escchr ttyset escape

The FLEX "ttyset" parameters allow the direct determination and specification of certain parameters of the console. Under OS/9, similar parameters are available, but determination of them must be done thru the "os9 i\$setStt" call and specification of them must be done thru the "os9 i\$setStt" call.

s drn system drive number
The FLEX system drive number is often used to determine the drive from which to load executable programs. Under OS/9, the system drive number corresponds to the default execution directory.

w drn working drive number
The FLEX working drive number is often used to determine the drive from which to load and save data files. Under OS/9, the working drive number corresponds to the default data directory.

sysmon system month
sysday system day
sysyer system year
The FLEX month, day, and year allow the direct determination and specification of the system date. Under OS/9, similar parameters are available, but determination of them must be done thru the "os9 r\$time" call and specification of them must be done thru the "os9 r\$time" call.

lstrm last terminator
cbuip line buffer pointer
curchr current character
prevchr previous character
These FLEX locations are associated with scanning of the command line buffer. Assuming the buffer has been defined as described above, these locations have been properly defined, and the OS/9 routines which process it have been defined as described earlier, these locations should be usable in the same manner under OS/9 as under FLEX.

errtyp error type
The error type contains the return code from the last FMS call. If it is used in a FLEX program, it should be replaced by a reference to the error code in the appropriate FCB.

loflag special I/O flag
FLEX programs use this location to control certain parameters of the console (specifically, to ignore the "ttyset" width and escape parameters if the location contains a non-zero value). Under OS/9, control of similar parameters may be performed (if it is necessary) thru the "os9 i\$setStt" and "os9 i\$setStt" calls. However, since this location is primarily used to help control printer devices, such parameter modification may be unnecessary since OS/9 controls the printer separately from the console.

outsut output switch
insut input switch
foaddr file output address
fiaddr file input address
filecho file input echo flag
These locations are used by FLEX to control the redirection of console output to and from the console or disk files. Since OS/9 redirection is controlled from the command line, the locations cannot be used in the same manner under OS/9.

memend memory end
This location indicates the end of the application program address space under FLEX. Since memory allocation is performed dynamically by OS/9, any usage of this location must be carefully evaluated, modified, and eliminated.

cpu typ cpu type flag
This location is logically composed of the following flags (MSB to LSB):
cpu 2mhz 2 mhz CPU clock rate
cpu slow Memory stretch active
cpu 50mhz 50 Mhz power line frequency
cpu ram CPU RAM is available
cpu tck 6819 real time clock available
cpu lobj I/O set up like old box
cpu time 6840 timer available
cpu xmem Extended memory active
The only significant flag in this location (with respect to OS/9) is the "cpu 2mhz" flag, which is often used to help determine timing programmatic delay loops. Since the OS/9 system is interrupt-driven, all delay loops must be carefully evaluated for possible modification.

ulcflg upper/lower case flag

This location controls the mapping of FLEX file names from lower case to upper case or not. Case is insignificant in OS/9 path names.

verify FMS verify flag

This location controls whether disk writes will be verified or not. Since OS/9 path descriptors determine whether or not disk writes will be verified, and the environment is so different, references to "verify" should probably be dropped.

Although they are not strictly FLEX storage locations, any access to the I/O and DAT addresses must be carefully reviewed for possible modification. Under OS/9 Level 1, the I/O and DAT addresses are available for access by user programs. However, timing loops will be disrupted by the interrupt-driven nature of OS/9. Any direct access to the console port must be reviewed because OS/9 programs the port for data ready interrupts; OS/9 may read the data from the port before the program is able to read it.

Direct access to the I/O and DAT addresses is impossible under OS/9 Level 2, and must be replaced by references to standard or custom-written device drivers. This replacement is also possible, and sometimes necessary, for complex device-handling requirements, under OS/9 Level 1. Such device drivers are beyond the scope of this article, but are described in the OS/9 System Programmer's Manual.

EXAMPLES

CENTRONICS RECEIVER

The first example shows a rather simple conversion of a program which inputs data from a parallel port and sends it to the FLEX console output device, which, of course, may then be redirected to disk thru the use of the "o" command.

The FLEX version of this program is as follows:

```
* centronics receiver for file xfer
                                opt    pag
C003  varms    equ    $c003    flex varms start
C018  outpt    equ    $c018    flex put character
E030  place    equ    $e030    pla address
0002  plaab    equ    $02      offset a=$00,b=$02
E004  eclac    equ    $e004    eclac address
C100
C100 7E E033    start    cir    place+plaab+1 address ddr
C103 7E E032    cir
C106 86 34      ldr      $34    c2 out manual low
C108 87 E033    sta      place+plaab+1 program it
C10A 86 E033    next    ldr      place+plaab+1 check for edge
C10E 2B 09      bmf      data    yes, read it
C110 86 E004    ldr      eclac    check eclac
C113 44        isra
C114 24 F5      bcc      next    no, loop
C116 7E C003    jmp      varms    exit to fflex
C119 86 E032    data    ldr      place+plaab get data
C11C 87 E032    sta      place+plaab reset
C11F 06 3C      ldr      $3C    c2 out manual high
C121 F7 E033    stb      place+plaab+1
C124 06 34      ldr      $34    c2 out manual low
C126 F7 E033    stb      place+plaab+1
C129 84 7F      anda     $7F    mask parity
C12B 81 0D      cmpa     $0D    cr
C12D 27 08      beq      output
C12F 81 20      cmpa     $20    sp
C131 25 08      blo      next    ignore other controls
C133 81 7F      cmpa     $7F    del
C135 27 04      beq      next    ignore dels
C137 8D C018    output   jsr      outpt    output to flex
C13A 20 CF      bra      next    go back for more
                                ead      start
```

An OS/9 version of this same program is as follows:

```
                                nam      centronics receiver
                                use      for os/9
                                * centronics receiver for file xfer to os/9
                                mod      /d0/dets/detsfile
0000 87C0006F    endmod,nam,orgmobjct,rsent+1,
                                start,endmod
0000          stack    rmb      256    stack space
0100          endmem   equ      *      no data space
0000 46696C69    name     fcs      "flexfer"
E030          place    equ      $e030    pla address
0002          plaab    equ      $02      offset a=$00,b=$02
E004          eclac    equ      $e004    eclac address
0019 7FE033      start    cir      place+plaab+1 address ddr
001B 7FE032      cir
001B 8634        ldr      $34    c2 out manual low
001D 87E033      sta      place+plaab+1 program it
0020 86E033      next    ldr      place+plaab+1 check for edge
0023 290C 0031   bmf      data    yes, read it
0025 86E004      ldr      eclac    check eclac
0028 44          isra
0029 24F5 0020   bcc      next    no, loop
002B C00000      ldr      $0000    normal exit
002E 103F06      os9       f$exit
0031 86E032      data     ldr      place+plaab get data
0034 87E032      sta      place+plaab reset
0037 C63C        ldr      $3C    c2 out manual high
0039 F7E033      stb      place+plaab+1
003C C634        ldr      $34    c2 out manual low
003E F7E033      stb      place+plaab+1
0041 847F        anda     $7F    mask parity
0043 810D        cmpa     $0D    cr
0045 2714 009B   beq      output
0047 8120        cmpa     $20    sp
0049 2503 0020   blo      next    ignore other controls
```

```

0040 817F      cmpa  $57f      drl      ignore dals
0040 27D1 0020    beq  next      cr
0040 8100      cmpa  $500
0051 2708 003B    beq  output    sp
0053 8120      cmpa  $520      ignore other controls
0055 25C9 0020    blo  next      del
0057 817F      cmpa  $57f      ignore dals
0059 27C3 0020    beq  next      output to os/9
007B 3402      output pshs  a
0090 1F41      tfr  s,x
009F 8601      lda  $501
0061 10BE 0001    ldy  $50001
0065 103F 8A      os9  lserit
0068 3502      puls  a
006A 20B4 0020    bra  next      go back for more
006C 66F6C5      word  *
006F      endword nqu
0000      and

```

Note that this conversion is valid only for OS/9 Level 1. It is not possible for a user program to access the I/O ports directly under OS/9 Level 2. However, in its limited environment, the techniques used in this converted program are valid and work quite well. It is not necessary to define device drivers and to use the other facilities of OS/9 Level 1 to access the I/O ports in a simple manner.

FILE-HANDLING CONVERSION

The second example demonstrates many of the conversions discussed earlier, primarily in terms of the file-handling aspects of FLFX. The program itself came from Leo Taylor's collection.

The FLFX version of this program is as follows:

opt pag

- * upper to lower case text converter
- * this program will convert an input file of any
- * case to an output file of lower case with the
- * following character sequences set to upper
- * case:
- * 1. any output character following a period, a
- * question mark, or an exclamation point will be
- * in upper case. spaces, carriage return, or line
- * feeds between the punctuation and character
- * will be ignored.
- * 2. an 'i' will be upper case if preceded by a
- * space and followed by either another space or
- * an apostrophe.
- * 3. the first character of a file will be
- * upper case.
- * program is called by: lowerc oldfile newfile.
- * program converted by leo taylor from lsl-11
- * program written by robert malster.
- * flex address assignments:

```

C000 flex equ $C000
C840 fcb equ flex+$840 file control block
C003 verms equ flex+$d03
C015 getch equ flex+$d15
C018 putchr equ flex+$d19
C01E pstrng equ flex+$d1e
C024 pchr1 equ flex+$d24
C020 getfll equ 'l' + 2d
C033 setext equ flex+$d35
C03F rpterr equ flex+$d3f
0403 fmscis equ flex+$1403
0406 fms equ flex+$1406

```

C100

org flex+\$100

* program starts by opening two files

```

C100 20 02 lowerc bra lower1
C102 01 vn fcb 1 version number
C103 04 flag fch p flag starts with period
0001 a equ 1 alpha bit of flag
0002 i equ 2 'i' bit
0004 p equ 4 'l' or 'i' or 'j'
C104 8C C840 lower1 ldx #fcb point to fcb
C107 8D C020 jsr getfll get the file name
C10A 25 0C bcs error
C10C 86 01 ldrna 0,1 set for read
C10E A7 R4 staa 0,x save in fcb
C110 8D C033 jsr setext set default ext
C113 8D D406 jsr fms open for read
C116 27 09 beq lower2
C118 8D C03F error jsr rpterr report error
C11B 8D D403 jsr fmscis close all files
C11E 7E C003 warmsl jmp warms return to flex
C121 8E C10F lower2 ldx #fcb2 point to fcb
C124 8D C020 jsr getfll get file name
C127 25 EF bcs error
C129 86 01 ldrna #1 set extension

```

```

C12B 8D C033 jsr setext
C12E 86 02 ldrna #2 set for write
C130 A7 R4 staa 0,x
C132 8D 0406 jsr fms open for write
C135 26 E1 bne error
C137 8E C840 looplow ldx #fcb point to read
C13A 8D 0406 jsr fms get character
C13D 27 14 beq readchk
C13F A6 01 ldrna 1,x check error
C141 81 08 cmpa #8 is it not?
C143 26 D3 bne error
C145 8E C10F ldx #fcb2 point to write
C148 86 04 ldrna #4 set for close
C14A A7 R4 staa 0,x
C14C 8D 0406 jsr fms close write file
C14F 26 C7 bne error
C151 20 C8 bra warmsl return to flex
C153 81 41 readchk cmpa #541 upper case?
C155 25 06 blo lower1
C157 81 5A cmpa #57a
C159 22 02 bhl lower1
C15B 8B 20 adda #520 make lower

```

* all characters lower case
* now check flag

```

C150 F6 C103 lower1 ldrna #flag
C160 C5 0C bltb #0 test char period type?
C162 26 24 hne upper force upper case
C164 C5 02 bltb #1 'i' just received?
C166 26 0F bne get1
C168 81 69 cmpa #569 lower case 'i'
C16A 26 2E bne noflag
C16C C5 01 bltb #0 test char alpha?
C16E 26 2A bne noflag
C170 CA 02 orab #1 set 'i' flag
C172 F7 C103 stab flag
C175 20 C0 bra looplow get next char
C177 81 20 get1 cmpa #520 space or control?
C179 2F 04 blo lspace
C17B 81 27 cmpa #' ' apostrophe?
C17D 26 06 bne not1
C17F 34 02 lspace psha
C181 86 49 ldrna #549 save present char
C183 20 04 bra send1
C185 34 02 not1 psha save present char
C187 86 69 ldrna #569 lower case 'i'
C189 8D 18 bsr write
C18B 35 02 puls restore char
C18D 5F C0 cfrb normal char
C18E 20 0A bra noflag treat normally
C190 81 61 upper cmpa #561 lower case?
C192 20 06 blt noflag
C194 81 7A cmpa #57a lower case?
C196 2E 02 bgt noflag greater than lc 2
C198 8D 20 suba #520 convert to upper
C19A F7 C103 stab flag
C19C 8D 00 bsr setflag set flags for this char
C19E 8D 02 bsr write
C1A1 20 94 bra looplow

```

* conversion done so write character

```

C1A3 8E C10F write ldx #fcb2 point to write
C1A6 8D 0406 jsr fms write character
C1A9 26 A4 bne error1
C1AB 39 rts

```

* subroutine to set flag byte

```

C1AC F6 C103 setflag ldrna #flag start with old flag
C1AF 81 20 cmpa #520 printing char?
C1B1 2E 04 bgt set11 yes process
C1B3 C4 0E andb #57e clear alpha
C1B5 20 02 bra setfll
C1B7 C4 F8 andb #57b printing char
C1B9 81 2E setfll cmpa #'.' period?
C1BB 27 08 beq setf2
C1BD 81 3F cmpa #'?' question mark?
C1BF 27 04 beq setf2
C1C1 81 21 cmpa #'!' exclamation point?
C1C3 26 02 bne setf3
C1C5 CA 04 orab #p set period type
C1C7 81 41 cmpa #541 alpha?
C1C9 20 10 blt setf5 upper case?
C1CB 81 5A cmpa #57a
C1CD 2F 0A blo setf4 alpha?
C1CF 81 61 cmpa #561
C1D1 7D 08 bit setf5
C1D3 81 7A cmpa #57a lower case alpha?
C1D5 2E 04 bgt setf5
C1D7 CA 01 setf4 orab #a
C1D9 C4 F8 andb #57b set alpha
C1DB F7 C103 stab flag clear p flag
C1DE 39 rts new flag bits
C1DF fcb2 rmb 320
end lowercase

```

An OS/9 version of this same program is as follows:


```

nam      lowercase conversion
use      /d0/dels/delsfile

* upper to lower case text converter
* this program will convert an input file of any
* case to an output file of lower case with the
* following character sequences set to upper case:
* 1. any output character following a period, a
* question mark, or an exclamation point will be
* in upper case. spaces, carriage return, or line
* feeds between the punctuation and character
* will be ignored.
* 2. an 'i' will be upper case if preceded by a
* space and followed by either another space or
* an apostrophe.
* 3. the first character of a file will be
* upper case.
* program is called by: lowerc oldfile newfile.
* program converted by leo taylor from isl-11
* program written by robert malster.

```

```

0000 87 CD 00 F9      mod      endmod,namend,Prgram=Objct,ReEnt,
                        start,endasm

```

* program equates declared before use

```

0001  vn      equ      1      version number
0001  a      equ      1      alpha bit of flag
0002  i      equ      2      'i' bit
0004  p      equ      4      'i' or 'j' or 'l'

```

* program storage

```

0000      stasms equ      *      start of data area
0000      flag  rmb      1      character flag

0001      fcb  rmb      320     file control blocks
0141      fcb2 rmb      320
0281      labuff rmb      128   line buffer
0301      lstrm  rmb      1      last terminator
0302      cbufl  rmb      2      line buffer pointer
0304      curchr rmb      1      current character
0305      prevch rmb      1      previous character

0306      endasm equ      *      end of data area

0000 4C 6F 77 72      namend fcs      "Lowercase"

0015      start equ      *      starting address

0015 34      30      lowerc pshs x,y      clear data area
0017 30 C4      lvar      stasms,u      clear data area
0019 108E 0306  ldy      #endasm-stasms
0010 0F 80      clr      ,x      -S01,y
001F 31 3F      lvar      clear
0021 26 FA 0010  bne      clear
0023 35 30      puls      x,y

0025 17 0001 00F9  lbr      initial      initialize line buffer
0028 86 04      lds      #p      period
002A A7 C4      sta      flag,u      flag starts with period

```

* program starts by opening two files

```

002C 30 41      lower1 leax fcb,u      point to fcb
002E 17 00C8 00F9  lbr      getfil      get the file name
0031 25 0A 003D  bcs      error
0033 86 01      lds      #1      set extension
0035 17 00C1 00F9  lbr      setext      set default ext
0038 17 00B8 00F9  lbr      fms01      open for read
003B 27 09 0046  bnq      lower2
003D 17 00B9 00F9  error  lbr      rpterr      report error
0040 17 00B6 00F9  lbr      fmsc1s      close all files
0043 16 00B3 00F9  varms1 lbr      return
0046 30 C9 0141  lower2 leax fcb2,u      point to fcb
004A 17 00AC 00F9  lbr      getfil      get file name
004D 25 EE 003D  bcs      error
004F 86 01      lds      #1      set extension
0051 17 00A5 00F9  lbr      setext
0054 17 00A2 00F9  lbr      fms02      open for write
0057 26 E4 003D  bne      error
0059 30 41      looplo leax fcb,u      point to read
005B 17 00B8 00F9  lbr      fms00      get character
005E 27 11 0071  bnq      readok
0060 A6 01      lds      1,x      check error
0062 81 03      ccmp      #E5E0F      is it eof?
0064 26 D7 003D  bne      error
0066 30 C9 0141  leax fcb2,u      point to write
006A 17 00AC 00F9  lbr      fms04      close write file
006D 26 CE 003D  error1 bne      error
006F 20 02 0043  bra      varms1      return

0071 81 41      readok ccmp #S41      upper case?
0073 25 06 007B  blo      lower1
0075 81 9A      ccmp      #S5a
0077 22 02 007B  bhl      lower1
0079 8B 20      adda      #S20      make lower

```

* all characters lower case
* now check flag

```

007B E6 C4      lower1 ldrb flag,u
007D C3 04      fcb  bltb #p      last char period type?
007F 26 7B 00AC  bne      upper      force upper case
0081 C3 02      fcb  bltb #1      'i' just received?
0083 26 0E 0093  bne      geti      lower case 'i'
0085 81 69      ccmp      #S69
0087 26 20 0086  bne      nolflag
0089 C3 01      fcb  bltb #a      last char alpha?
008B 26 29 0086  bne      nolflag
008D CA 02      orb      #1      set 'i' flag
008F E7 C4      stb      flag,u
0091 20 C6 0059  bra      looplow      get next char

0093 81 20      geti      ccmp      #S20      space or control?
0095 26 04 009B  blo      lspace
0097 81 27      ccmp      #' '      apostrophe?
0099 26 06 00A1  bne      notl
009B 34 02      lspace pshs a      save present char
009D 86 49      lds      #S49
009F 20 04 00A5  bra      sendi

00A1 34 02      notl      pshs a      save present char
00A3 86 69      lds      #S69      lower case l
00A5 8D 17 00BE  sendi      bsr      write
00A7 33 02      puls      a      restore char
00A9 5F          clrb      normal char
00AA 20 0A 0086  bra      nolflag      trust normally

00AC 81 61      upper  ccmp      #S61      lower case?
00AE 20 06 0086  blt      nolflag
00B0 81 7A      ccmp      #S7a      lower case?
00B2 2E 02 0086  bgt      nolflag      greater than lc i
00B4 80 20      suba      #S20      convert to upper
00B6 E7 C4      nolflag stb      flag,u
00B8 8D 0E 00C8  bsr      setflag      set flags for this char
00BA 8D 02 00BE  bsr      write
00BC 20 9B 0059  bra      looplow

```

* conversion done so write character

```

00BE 30 C9 0141  write leax fcb2,u      point to write
00C2 17 0034 00F9  lbr      fms00      write character
00C5 26 A6 006D  bne      error1st
00C7 39          rts

```

* subroutine to set flag byte

```

00C8 E6 C4      setfla ldrb flag,u      start with old flag
00CA 81 20      ccmp      #S20      printing char?
00CC 2E 04 00D2  bgt      setfl1      yes process
00CE C4 FE      andb      #S1e      clear alpha
00D0 20 02 00D4  bra      setfl1
00D2 C4 FB      setfl1 andb      #S1b      printing char
00D4 81 2E      setfl1 ccmp      #1      period?
00D6 27 08 00E0  beq      setfl2
00D8 81 3F      ccmp      #1?      question mark?
00DA 27 04 00E0  beq      setfl2
00DC 81 21      ccmp      #1!      exclamation point?
00DE 26 02 00E2  bne      setfl3
00E0 CA 04      setfl2 orb      #p      set period type
00E2 81 41      setfl3 ccmp      #S41      alpha?
00E4 2D 10 00F6  blt      setfl5
00E6 81 5A      ccmp      #S5a      upper case?
00E8 2F 08 00F2  blo      setfl4
00EA 81 61      ccmp      #S61      alpha?
00EC 2D 08 00F6  blt      setfl5
00EE 81 7A      ccmp      #S7a      lower case alpha?
00F0 2E 04 00F6  bgt      setfl5
00F2 CA 01      setfl4 orb      #a      set alpha
00F4 C4 FB      andb      #S1b      clear p flag
00F6 E7 C4      setfl5 stb      flag,u      new flag bits
00F8 39          rts

```

* the following entry points would be replaced
* by the code described in an earlier section;
* it is not included here to avoid repetition.

```

00F9      initlo equ      *      initialize line buffer
00F9      varms equ      *      return to flex
00F9      getfil equ      *      get file specifications
00F9      setext equ      *      set file extension
00F9      rpterr equ      *      report error

00F9      fmsc1s equ      *      close files
00F9      fms00 equ      *      get/put next character
00F9      fms01 equ      *      open input
00F9      fms02 equ      *      open output
00F9      fms04 equ      *      close

00F9      endmod equ      *
0015      end      lowercase

```

BUFFERED SEQUENTIAL I/O

The third example demonstrates a scheme for implementing buffered sequential I/O using OS/9 system requests, with the goal of optimizing program performance in many I/O-bound programs.

Since every call to OS/9 from a module executing under its control causes an interrupt, and interrupts contribute heavily to system overhead, it would seem logical to attempt to significantly reduce the

number of OS/9 calls in order to reduce the system overhead required to process a module.

In I/O-bound portions of modules involving significant amounts of sequential I/O, the easiest manner in which to reduce the number of calls to OS/9 is to request that more than one byte at a time be transferred between the module and OS/9 during sequential I/O operations. Every OS/9 call not performed is at least 35 machine cycles saved, just for the "wait" and "return" instructions, not counting the OS/9 overhead, and the savings may be great.

The skeletal module listed below contains several potentially useful routines which implement a form of buffered sequential I/O under OS/9. They are loosely based upon an earlier set of similar routines provided in TSC's Uniflow Programmer's Guide, and are used heavily in CSC products to increase performance.

The routines in the sample module below are as follows:

initbuf Initialize buffers
demonstrates how to initialize buffer control areas;
re-opening files requires reinitializing these;
getbuf get character from buffer
returns next character, reloading buffer if needed;
putbuf put character into buffer
outputs next character, dumping buffer if needed;
clsbu close output buffer
dumps used portion of output buffer if not empty;
must be called before output file is closed.

Associated with the routines is a control area for each buffer. The comments associated with the definitions of the offsets indicate the usage of each field in each control area. The first few items in each control area are analogous to items in the FLEX CDB, and are indicated as such. Thus, these routines could fit in well in the case of programs being converted from FLEX to the OS/9 file environment.

```

                                buffered I/O routines
                                /d0/dels/delsfile
0000 87 00 01 4C      nam      use
                        mod      endmod,named,PrgrmObjct,ReEnt,
                        start,endmem

**
0000      buffb equ 0      FCB function code
0 1      buffer equ 1     FCB error code
0 2      buffac equ 2     FCB activity status
0003      buffpn equ 3     FCB path number
0004      buffre equ 4     edited/raw function code
0005      buffbp equ 5     buffer pointer
0007      buffcp equ 7     character pointer
0009      buffbl equ 9     buffer length
000B      buffnc equ 11    character counter
**
0010      conlen equ 16    length of control area
0020      fnmlen equ 32    length of path name
0100      buflen equ 256   length of buffer
**
0001      read equ 1       read buffer raw
0002      rwnltn equ 2     read buffer edited
0003      wrlln equ 3      write buffer raw
0004      wrtln equ 4      write buffer edited
**
0000      stamem equ *     start of data area
**
0000      inpath rdb fnmlen input path name
0020      incont rdb conlen input path controls
0030      inbuff rdb buflen input buffer
0130      otpath rdb fnmlen output path name
0150      otcont rdb conlen output path controls
0160      otbuff rdb buflen output buffer
0260      ticont rdb conlen terminal input controls
0270      libuff rdb buflen terminal input buffer
0370      tocont rdb conlen terminal output controls
0380      tobuff rdb buflen terminal output buffer
**
0480      endmem equ *     end of data area
**
0000 42 75 66 66      named fcs "Buffer"
**
0013      start equ *     starting address
**
0013 30 C4      leax stamem,u clear buffers
0015 10RE 0480      ldx $(endmem-stamem)
0019 6F 80      clr x
001B 31 3F      leay -$(01,y
001D 26 FA 0019      bne clear
**
001F 17 00C7 00E9      lbr initbuf initialize buffers
**
** program logic goes here
**
0022 103F 06      os9 f$Evlt exit program
**
*****
* getbuf gets character from buffer
* requires x=buffer pointer
* returns cc if no error
* cs if error, b=error
* vs if eof
* if all OK, a=character
0025 34 64      getbuf pshs b,y,u
0027 EC 0B      ldd buffnc,v characters in buffer

```

```

0029 26 20 0058      bne getbne
002B A6 03      ldx buffpn,v path number
002D 10AE 09      ldy buffbl,v buffer length
0030 E6 04      ldb buffre,v function code
0032 54 10      pshs x
0034 AE 05      ldx buffbp,v buffer address
0036 C1 02      cmpb #readln
0038 26 05 003F      bne getbrw
003A 103F 8B      os9 l$Readln
003D 20 0A 0049      bre getbrd
003F 4D      getbrw fata
0040 26 04 0046      bne getbrt
0042 108E 00D1      ldy f$D0D1
0044 103F 8B      getbrt os9 l$Read
0049 35 10      getbrd puls x
004B 25 1F 006C      bcs getber
004D 10AF 0B      sty buffnc,v update character counter
0050 2F 16 0068      bnq getbel
0052 EC 05      ldd buffbp,v check for eof
0054 E0 07      std buffcp,v reset character pointer
0056 EC 0B      ldd buffnc,v update character counter
0058 83 0001      getbne subd f$D0D1
005B E0 0B      std buffnc,v count characters
005D 10AE 07      ldy buffcp,v
0060 A6 A0      ldx y
0062 10AF 07      sty buffcp,v update character pointer
0065 5F      clrb cc,v
0066 35      puls b,y,u,pc
0068 1A 02      getbel orcc f$02
006A 35 04      puls b,y,u,pc
006C C1 03      getber cmpb #f$03
006E 27 F8 0068      beq getbel
0070 1A 01      orcc f$01
0072 32 61      leos f$01,s
0074 35 E0      puls y,u,pc
**
* putbuf puts character into buffer
* requires x=buffer pointer
* a=character
* returns cc if all OK
* cs if error, b=error
**
0076 34 66      putbuf pshs d,u,y
0078 EC 0B      ldd buffnc,v characters in buffer
007A 26 21 009D      bne putbne
007C A6 03      ldx buffpn,v path number
007E F6 04      ldb buffre,v function code
0080 10AE 09      ldy buffbl,v buffer length
0083 34 10      pshs x
0085 AE 05      ldx buffbp,v buffer address
0087 C1 04      cmpb #wrtln
0089 26 05 009D      bne putbrw
008B 103F 8C      os9 l$Write
008E 20 03 00A3      bre putbrt
0090 103F 8C      putbrw os9 l$Write
0093 35 10      putbrt puls x
0095 25 17 00AE      bcs putber
0097 EC 05      ldd buffbp,v check for error
0099 E0 07      std buffcp,v update character pointer
009B EC 0B      ldd buffbl,v update buffer length
009D 83 0001      putbne subd f$D0D1
00A0 E0 0B      std buffnc,v count chars
00A2 10AE 07      ldy buffcp,v update character count
00A5 35 02      puls a
00A7 A7 A0      sty y
00A9 10AF 07      sty buffcp,v update character pointer
00AB 35 A4      puls b,y,pc
00AD 35 20      putber puls y
00B1 1A 01      orcc f$01
00B3 35 E0      puls y,u,pc
**
* clsbu closes output buffer
* requires x=buffer pointer
* returns cs if error, b=error
**
00B5 34 66      clsbu pshs d,u,y
00B7 EC 0B      ldd buffbl,v characters in buffer
00B9 A3 0B      subd buffnc,v
00BB 27 1E 00DB      beq clsbem
00BD 1F 02      tfr d,y
00BF A6 03      ldx buffpn,v character count
00C1 E6 04      ldb buffre,v path number
00C3 34 10      pshs x
00C5 AE 05      ldx buffbp,v buffer address
00C7 C1 04      cmpb #wrtln
00C9 26 05 00D0      bne clsbw
00CB 103F 8C      os9 l$Write
00CE 20 03 00D3      bra clsbu
00D0 103F 8C      clsbw os9 l$Write
00D3 35 10      clsbu puls x
00D5 25 0B 00E2      bcs clsber
00D7 EC 05      ldd buffbp,v check for error return
00D9 E0 07      std buffcp,v update character pointer
00DB EC 0B      clsbem ldd buffbl,v
00DD E0 0B      std buffnc,v buffer size
00DE 4F      clrb cc,v
00E0 35 E6      puls d,y,u,pc
00E2 35 20      clsber puls y
00E4 4F      puls y
00E5 1A 01      orcc f$01
00E7 35 E0      puls y,u,pc
cs, error
cs, b=error

```

```

**
* Inbuf initializes buffers
**
00E9 06 01 Inbuf lde #read set up buffers
00EA A7 C8 24 sta lncnt+buffre,u
00EB A7 C9 0264 ste f1cnt+buffre,u
00EC 06 03 lde #write
00ED A7 C9 0154 ste ofcnt+buffre,u
00EE 06 04 lde #rfile
00EF A7 C9 0374 sta f1cnt+buffre,u
00F0 CC 0100 ldd #buffin
00F1 ED C9 0269 std f1cnt+buffbl,u
00F2 ED C9 0379 std f1cnt+buffbl,u
00F3 ED C9 0377 std f1cnt+buffcp,u
00F4 ED C8 29 std lncnt+buffbl,u
00F5 ED C9 0159 std ofcnt+buffbl,u
00F6 ED C9 0157 std ofcnt+buffcp,u
00F7 06 01 lde #s01
00F8 A7 C9 0373 stb f1cnt+buffcp,u
00F9 30 C8 30 leax lncnt+buffbl,u
00FA AF C8 25 stx lncnt+buffcp,u
00FB AF C8 27 stx lncnt+buffcp,u
00FC 30 C9 0160 leax ofbuff,u
00FD AF C9 0159 stx ofcnt+buffcp,u
00FE AF C9 0157 stx ofcnt+buffcp,u
00FF 30 C9 0270 leax f1buff,u
0100 AF C9 0265 stx f1cnt+buffcp,u
0101 AF C9 0267 stx f1cnt+buffcp,u
0102 30 C9 0380 leax f2buff,u
0103 AF C9 0375 stx f1cnt+buffcp,u
0104 AF C9 0377 stx f1cnt+buffcp,u
0105 39 rts

014C endmod equ *
0000 on

```

MEDIA CONVERSION

The physical conversion of programs from FLEX to OS/9 is made materially more difficult by the fact that neither can logically read or write the other's diskettes. Both use 256-byte data segments with (potentially) the same disk interface hardware. Unfortunately, the disk directory and identification sector formats are quite different and incompatible.

Until recently, the only reliable means of converting media from FLEX to OS/9 (or vice versa) has been the use of two systems with communications facilities, either physically hard-wired together or remote over modems.

The OF program, distributed by Data-Comp, partially solves this problem. It utilizes a split format diskette, which appears to FLEX to be its diskette, and to OS/9 to be its OS/9 diskette. A BASIC09 program provides the disk formatting and OS/9 interface, whereas standard FLEX utilities provide the necessary FLEX interface. Using this technique, two systems (or one system alternating between FLEX and OS/9) are still required to perform the conversion of programs and data between the operating systems. Arbitrary FLEX diskettes may still not be directly converted to OS/9, and vice versa. However, within the restrictions just discussed, OF is quite effective, and a great improvement over modems.

SUMMARY

This article attempted to provide a structure for the conversion of application programs intended for execution under the FLEX operating system to operation under the OS/9 operating system.

It discussed the additional requirements on programs running under OS/9, and provided equivalents for many FLEX entry points and storage locations commonly used by application programs.

It provided two examples of program conversions and one example of a program optimization scheme intended to enhance the performance of converted I/O-bound programs.

It discussed the problems of transporting the program and data files from OS/9 to FLEX and suggested several solutions.

With this information, a programmer generally familiar with assembler languages for both OS/9 and FLEX should be able to readily convert programs from FLEX to OS/9.

BIT BUCKET

1.00-EDITOR'S NOTE: PLEASE NOTE THAT THIS PROGRAM
2.00-IS ALSO ON THE 68 MICRO JOURNAL BBS. THOSE NOT
3.00-WISHING TO TYPE IN THE CODE MAY DOWNLOAD FROM
4.00-THE BBS.

5.00-
6.00-ALSO PLEASE NOTE THAT THE SYMBOL '0' DENOTES THE
7.00-'ESC' FUNCTION.

8.00-
9.00-
10.00-
11.00-
12.00-
13.00-
14.00-
15.00-
16.00-
17.00-
18.00-
19.00-
20.00-
21.00-
22.00-
23.00-
24.00-
25.00-
26.00-

MODEMPROGRAM for ACIA 6850, CPU 6809.

and FLEX dos

Both ACIA's have to be wired for IRQ 1

JUNE 1982

Written by Esko Anttilainen (SHAKP)

and Avo Kesh I SHAKVO)

Modified extensively by Steven M. Ward on 07 FEB 83.

Version number change is from version 2 to version 3.

The following changes were made:

```

27.00-
28.00-
29.00-
30.00-
31.00-
32.00-
33.00-
34.00-
35.00-
36.00-
37.00-
38.00-
39.00-
40.00-
41.00-
42.00-
43.00-
44.00-
45.00-
46.00-
47.00-
48.00-
49.00-
50.00-
51.00-
52.00-
53.00-
54.00-
55.00-
56.00-
57.00-
58.00-
59.00-
60.00-
61.00-
62.00-
63.00-
64.00-
65.00-
66.00-
67.00-
68.00-
69.00-
70.00-
71.00-
72.00-
73.00-
74.00-
75.00-
76.00-
77.00-
78.00-
79.00-
80.00-
81.00-
82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

```

SYSTEM EQUATES

60.00+TERMOU EQU SC018 FLEX CHARACTER OUTPUT

61.00+WARNIS EQU SC003

62.00+PSTRNG EQU SC01E

63.00+MODCH EQU SC7E0

64.00+TERMIN EQU SC7E8

65.00+IRVEC EQU SC7C0

66.00+INBUF EQU SC01B

67.00+SETEX EQU SC033

68.00+DEFTL EQU SC02D

69.00+FWCLS EQU SC043

70.00+FMS EQU SC048

71.00+FCB EQU SC040

72.00+RPTERR EQU SC03F

73.00+PCRLF EQU SC024

74.00+MEMCH EQU SC02B

75.00+XCON EQU \$11

76.00+XOFF EQU \$13

77.00-
78.00-
79.00-
80.00-
81.00-
82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-
95.00-
96.00-
97.00-
98.00-
99.00-
100.00-
101.00-
102.00-
103.00-
104.00-
105.00-
106.00-
107.00-
108.00-
109.00-
110.00-
111.00-
112.00-
113.00-
114.00-
115.00-
116.00-
117.00-
118.00-
119.00-
120.00-
121.00-
122.00-
123.00-
124.00-
125.00-
126.00-
127.00-
128.00-
129.00-
130.00-
131.00-
132.00-
133.00-
134.00-
135.00-
136.00-
137.00-
138.00-
139.00-
140.00-
141.00-
142.00-
143.00-
144.00-
145.00-
146.00-
147.00-
148.00-
149.00-
150.00-
151.00-

80.00+START BRA START0

81.00+FCB FCB

82.00-
83.00-
84.00-
85.00-
86.00-
87.00-
88.00-
89.00-
90.00-
91.00-
92.00-
93.00-
94.00-

```

192.00= CHPA #30C
193.00= BEQ RELTOP3
194.00= BRA RELTOP
195.00=RELOOP2 CHPA #37F
196.00= BEQ RELOOP
197.00=RELOOP3 1ST EPLG
198.00= BEQ LOP
199.00= JSR TEROUT OUTPUT TO TERMINAL
200.00= CHPA #30
201.00= LOP
202.00= LDA #1A
203.00= JSR TEROUT OUTPUT A LINE FEED
204.00= LDA #3D
205.00= LOP
206.00= LOP INDCR1
207.00= ANDB #00000010
208.00= BEQ LOP NOT READY YET
209.00= WAIT WAIT
210.00= CHPA #300
211.00= BNE WAIT
212.00= STA INDCR1
213.00= BRA RELOOP
214.00=REIO15 LDX #WRITXT WRITE TEXT TO DISK
215.00= JSR PSTRNG
216.00= LOP CDISINT
217.00= STB IACIACR1 NOT TERMINAL IRQ
218.00= JSR INBUFF
219.00= LOP CENAINIT
220.00= STB IACIACR1 TERMINAL IRQ AGAIN
221.00= LDX #FCB
222.00= JSR GETPIL
223.00= LBCS ERROR
224.00= LDA #1
225.00= JSR SETEXT EXT=.TXT
226.00= LDA #2
227.00= STA .X
228.00= JSR FMS OPEN FOR WRITE
229.00= LBNB ERROR
230.00= LDT #BUFSTA
231.00=RELOOP CHPY POINTER
232.00= BEQ WSTOP
233.00= L A
234.00= JSR FMS WRITE TO DISK
235.00= LBNB ERROR
236.00= BRA MLOP
237.00= LDA #4
238.00= STA .X
239.00= JSR FMS CLOSE FILE
240.00= LBNB ERROR
241.00= LDX #BUFSTA
242.00= STX POINTER CLEAR BUFFER
243.00= JMP START1
244.00= LDA #3 MASTER RESET ACIA
245.00= STA IACIACR1
246.00= LDA CDISINT
247.00= STA IACIACR1 SETUP NORMAL CONSOLE CONFIGURATION *****
248.00= LDX IROVSA
249.00= STX IROVEC RESET SYSTEM IRQ
250.00= JMP WARMPS
251.00=ESCA JSR FMSCLS CLOSE OPEN FILES
252.00= LDA #3FF
253.00= STA SAVFLG DO NOT SAVE TEXT ON RECEIVE
254.00= LDA #300
255.00= STA WAITPL
256.00= LDX #BUFSTA
257.00= STX POINTER RESET TEXT POINTER
258.00= JMP START1 JUMP TO NORMAL RECEIVE ROUTINE
259.00=REINTER CHP INDCR1
260.00= BPL CHPIN
261.00= LDA INDCR1
262.00= ANDA #37F STRIP PARITY
263.00= CHPA #30FF
264.00= RNE INTER2
265.00= LOP #3FF
266.00= STB #A11PL
267.00= RTI
268.00=REINTER2 CHPA #XON
269.00= BNE LINEF
270.00= LOP #300
271.00= STB WAITPL
272.00= RTI
273.00=RELINEF CHPA #30A
274.00= BNE BELL
275.00= JSR TEROUT
276.00= RTI
277.00=REBELL CHPA #307
278.00= BNE INTER3
279.00= JSR TEROUT
280.00= RTI
281.00=REINTER3 CHPA #30C
282.00= BEQ INTER3
283.00= CHPA #30C
284.00= BEQ INTER3
285.00= CHPA #311
286.00= BEQ INTER4
287.00= RTI
288.00=REINTER4 CHPA #37F
289.00= BNE INTER3
290.00= RTI
291.00=REINTER5 JSR TEROUT
292.00= ST SAVFLG
293.00= BNE RII
294.00= LDX POINTER
295.00= CHPX MEMEND MEMORY FULL
296.00= BEQ WRISTO
297.00= STA .X
298.00= STX POINTER
299.00= BRA INTER
300.00=REWRISTO LDX #FULTXT MEMORY FULL
301.00= JSR PSTRNG
302.00= LDA #START1 NEW RETURNADDRESS
303.00= STX 10.5
304.00= RTI
305.00=RECHIN LOP IACIACR1 CHARACTER INPUT HANDLER
306.00= BPL RTI
307.00= LDA IACIARE GET CHAR
308.00= ANDA #37F STRIP PARITY
309.00= CHPA RETUR RETURN TO FLEX ?
310.00= BEQ RET
311.00= CHPA ECHO ECHO ON/OFF ?
312.00= BEQ EXOSET
313.00= CHPA SAVE SAVE TEXT ?
314.00= BEQ SAYSET
315.00= CHPA TRANSM TRANSMIT TEXT ?
316.00= BEQ TRAMIT
317.00= CHPA WRITE WRITE TO DISK ?
318.00= BEQ DISWR1

```

```

278.00= CHPA ESCAPE START ALL OVER AGAIN ?
279.00= BEQ ESCAP
280.00= TEST ECHO 12
281.00=RECHIN LOP
282.00= JSR TEROUT CHARACTER TO TERMINAL
283.00= LOP INDCR1
284.00= ANDB #00000010
285.00= BEQ OUT
286.00= STA INDCR1
287.00= RTI
288.00= RET LDX #EXIT
289.00= STX 10.5 NEW RETURNADDRESS
290.00= RTI
291.00= EXOSET CHPA EFLG
292.00= RTI
293.00= SAYSET CLR SAVFLG
294.00= LDX #SAVTXT
295.00= JSR PSTRNG
296.00= JSR PCRLF
297.00= LDX #BUFSTA
298.00= STX POINTER CLEAR THE BUFFER
299.00= RTI
300.00= TRAMIT LDX #DISFIL
301.00= STX 10.5 NEW RETURNADDRESS
302.00= RTI
303.00= DISWR1 LDX #WRIO15
304.00= STX 10.5 NEW RETURNADDRESS
305.00= RTI
306.00= ESCAP LDX #ESCA
307.00= STX 10.5 NEW RETURNADDRESS
308.00= LOP
309.00= ERROR LDA 1-X ERROR BITE
310.00= CHPA #B
311.00= BNE STOP
312.00= LDA #4
313.00= STA .X
314.00= JSR FMS CLOSE FILE
315.00= LDA 1-X
316.00= BNE STOP
317.00= LDX #REDTXT
318.00= JSR PSTRNG
319.00= JSR PCRLF
320.00= JMP START1
321.00= STOP JSR RPTERR
322.00= JSR FMSCLS
323.00= LDA #300
324.00= STA WAITPL
325.00= JMP START1
326.00= FULTXT FCC /Memory Full !!/
327.00= FCC $07,$04
328.00= DISTXT FCC /Name of diskfile to transfer ? /
329.00= FCC $04
330.00= REDTXT FCC /Transmission done/
331.00= FCC $07,$04
332.00= STATXT FCC $0,$A,$A
333.00= FCC /on = Transmit File/
334.00= FCC $0,$A
335.00= FCC /on = Save Text/
336.00= FCC $0,$A
337.00= FCC /on = Write Text to Diskfile/
338.00= FCC $0,$A
339.00= FCC /on = Echo On-Off Toggle (half or full duplex)/
340.00= FCC $0,$A
341.00= FCC /on = New; clear receive buffer and restart/
342.00= FCC $0,$A
343.00= FCC /on = Return to Flex/
344.00= FCC $0,$A,$A
345.00= FCC /on = In conversation mode; "Save Text" is Off/
346.00= FCC $0,$A,$A,$A
347.00= SAYTXT FCC /Saving text into buffer /
348.00= FCC $04
349.00= WRITXT FCC /Write saved text to disk. FILENAME ? /
350.00= FCC $04
351.00= BUFSTA RMB 1 START OF BUFFER
352.00= END

```



MICROWARE.

PRESS RELEASE

September 15, 1983
For Immediate Release
Contact: Andy Ball, 515-279-8844

RMA RELOCATABLE MACRO ASSEMBLER INTRODUCED BY MICROWARE

Microware has introduced RMA, a powerful new full feature relocatable assembler and linkage editor for the OS-9 Operating System. RMA was designed to process both manually-written and compiler-generated assembly language programs with special features specifically designed for the OS-9 environment.

Sections of assembly language programs can be independently assembled to "relocatable object files". The linkage editor takes any number of relocatable object files and/or library sections and combines them into a single relocatable OS-9 program. The linker can optionally generate a detailed load map listing.

RMA has special facilities for convenient generation of position-independent and relocatable programs. For example, global data variables (including indexed and direct addressing modes commonly used in OS-9 programs) and program references are automatically processed properly.

The macro facility permits commonly used instruction sequences to be defined once, then used within the program as often as desired with automatic parameter substitution. Conditional assembly features permit only specified sections of the program to be assembled. This can be used to produce various customized versions of a program for a single master source file.

Dear Don,

In the July 81 issue, you published my program called "RADCA", a small data management system. An oversight on my part, which was pointed out by a few readers in personal correspondence, is the inability of the program to handle parallel printers. Also included in the program was a non-standard I/O call to the Hilisim firmware monitor, which, not surprisingly, left a few systems cold in their tracks (no pun intended).

The above mentioned problems have now been cured, along with an additional enhancement which amounts to being able to redirect the output to either the terminal or printer on all functions. The printer cure amounts to loading (automatically), and vectoring all printer output through the PRINT.CVS drivers. Also the addition of being able to specify which drive the data files reside was added.

Another problem, which is not really a problem, but more of an irritation, is the fact that "RADCA" uses lowercase characters in the disk data file specifications. This provides a sort of file protection in that Flex normally does not recognize lowercase characters and makes the files virtually impossible to delete, if an attempt at deleting the data file is made. Flex will return a "Not Found" error. This problem (irritation) has not been resolved, however it can be cured by changing all occurrences of "Catalog" to "CATALOG" in the source file and re-assembling into a .COM file. While in the Flex editor (edit:cas RADCA), use the command sequence: `^PC/Catalog/CATALOG/100C`. Also, it may be advisable to eliminate the modification to Flex's setup in the beginning of the program.

Again, as in the original article, if anyone needs help, please feel free to contact me. Please include a disk (with other software you wish to share if possible), and sufficient return postage, and I will return the updated source file and an assembled .COM file.

Robert Lund
P.O. Box 806
Hillside, IL 60162

Robert M. Morrison
P.O. Box 912
Alb., N. 09617
Tel. (601) 496-0343

October 20, 1983

'68' Micro Journal
P.O. Box 849
Hixson, TN 37343

Dear Don,

I have some comments about the article entitled "SELECTIVE DISK" by Derek Giteleson in the September 1983 issue (Volume V, Issue IX) of '68' Micro Journal.

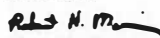
You may have to refresh my memory because it has been a long time since my system has been fully operational (due to a virus) in my home here to Frankfurt, Germany from (Atlanta City, Georgia). But I believe that all versions of FLEX include the DISK command. Although this command is not quite as versatile as the DISK command written by Mr. Giteleson, it does provide most of the capabilities that the author claims was lacking in FLEX. To use his example of DISK e.b., the same listing could be obtained by using the command `DIP .B`. his second example, DISK FILE.TXT could be simply DISK FILE.TXT using the FLEX provided DISK command.

My purpose in writing is definitely NOT to belittle Mr. Giteleson! On the contrary, his program has some very desirable additional features that do not exist in the normal DISK command: specifically, the ability to list all test files ending in CLOAK by using the command `DIP CLOAK.TXT`. In addition, he has added some very nice printer control features to make the printed output much more readable.

I mention this because of my hatred of time wasted spent re-inventing the wheel. This is one of the greatest benefits that I get from your magazine -- the exchange of ideas that occurs between the readers. I can only wonder if Mr. Giteleson would have spent the time writing DISK if he already had a copy of FLEX. I assume that he didn't know of its existence based on the first paragraph in his article. If so, I hope that it was for the few additional capabilities in DISK. I believe DISK to be a fine utility, but I think that \$4.95 is somewhat outrageous to pay for a copy of the source code. Even if DISK did not exist this would be an excessive amount to pay.

Along the line of protection duplication of work, I am working on converting SASCA 1 to 68C code. If anyone is interested in assisting in this project, or has already done some work in this area, please call or write me.

Sincerely,


Robert M. Morrison

Dear Folks;

I did a very foolish thing by letting my '68 MJ subscription lapse. Reading endless inane articles ("make a pretty snowflake with COCO!", "stop the deadly pizza's with Color Astro Chomp!") and gushing, simplistic, uninformed, and invariably dead-wrong reviews, I've come to appreciate Computer Publishing's Editorial Policies. I personally find 68 MJ more useful and informative than the C.C. dedicated magazines. But, then again, I haven't yet subscribed to Color M.J. Please find enclosed a check for \$40 (\$16.50+\$24.50) to start a subscription of Color M.J., and renew 68 MJ. This will probably get to you after you've shipped, but if you could begin these subscriptions with the current issues, GREAT!

Martin Maltby
Route 1, Box 41-B
Hartsburg, MO 65039

OS/9™, FLEX™, UNIFLEX™, IBM PC™ Software

SUPER SLEUTH DISASSEMBLER \$99-FLEX \$100-UNIFLEX \$101-OS/9

This program processes 6800/1/2/3/5/8/9 6502 programs, enabling the user to interactively analyze, modify, and disassemble (with labels) object code, with output to terminal, printer, and disk, and cross-reference and label definition capabilities. Object-Only for Color FLEX \$50, Color DOS \$49, Color OS/9 \$50.

Z-80/8080/5 SUPER SLEUTH DISASSEMBLER \$99-FLEX \$100-UNIFLEX \$101-OS/9

This version of SUPER SLEUTH processes Z-80/8080/5 object code on the 6800/1/9

CROSS-ASSEMBLERS each \$50 3/\$100-FLEX each \$60 5/\$120-UNIFLEX each \$55 3/\$110-OS/9

These programs and macros enable the user to process 6800/1, 6805, Z-80, 8080/5 programs in original format.

The TSC macro assembler is required for FLEX/UNIFLEX and the QSM assembler is required for OS/9.

[14]6805 and 6502 DEBUGGING SIMULATORS each \$75-FLEX \$80-UNIFLEX \$100-OS/9

These programs enable the user to interactively analyze, modify, and debug [14]6805 and 6502 object code.

6502-TO-6809 XLATOR SYSTEM \$75-FLEX \$80-UNIFLEX \$85-OS/9

This program enables the user to translate 6502 assembler code into 6809 assembler code, noting inexact conversions.

6800-6809 & 6809 PIC XLATORS both \$50-FLEX \$60-UNIFLEX \$75-OS/9

These programs enable the user to translate 6800/1 assembler programs to 6809 mnemonics and to convert 6809 programs to position-independent code and data, using PC, S, U, X, and Y as base registers.

OS/9 and UNIFLEX SIMULATORS FOR FLEX each \$100-FLEX

The programs enable the user to debug OS/9 and UNIFLEX assembler programs using the TSC DEBUG and other facilities of FLEX.

DISK UTILITY PROGRAMS all \$50-FLEX

These programs enable the user to list/modify the SIR, to edit sectors, to test entire diskettes, to linearize the free list, to back up one disk to another, etc.

FULL SCREEN FORMS DISPLAY (6809 X-BASIC) \$50-FLEX \$75-UNIFLEX \$60-IBMPC

These programs enable the user to define and generate table-driven full-screen display and data-entry programs.

FULL SCREEN MAILING LIST (6809 X-BASIC) \$100-FLEX \$110-UNIFLEX \$105-IBMPC

These programs enable the user to define and maintain mailing-list-oriented data bases.

FULL SCREEN INVENTORY/MRP (6809 X-BASIC) \$100-FLEX \$120-UNIFLEX \$110-IBMPC

These programs enable the user to define and maintain inventories, and include hierarchical materials requirement planning.

TABULA RASA SPREADSHEET (6809 X-BASIC) \$100-FLEX \$125-UNIFLEX

These programs enable the user to generate and maintain tabular compilation schemes, providing a simple user interface and sophisticated report generation, similar to DESKTOP/PLAN (IBM Desktop Computing).

5.25" DSDD SOFT-SECTORED DISKETTES \$1.50 each in 50's

with Type-A jackets, hub rings, write-protect tabs, and labels. ADD \$4.00 PER 50 FOR SHIPPING.

Programs in source on disk specify size, sides, density, type, computer, O/S.

Detailed printed manuals provided with all products.

For VISA and MASTERCARD give account, exp. date, phone, US funds only, add 5% (10% foreign for shipping).

Open Purchase Orders for D and B rated clients only. Call or write for catalog and dealer information.

*FLEX is a trademark of Technical Systems Consultants. *DSDD is a trademark of Microware.

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone Number 404-483-1717/4570

CLASSIFIED ADVERTISING

New, never installed or used - High Speed SWTPC A-1 Tape Back-up system, 20 megabyte capacity - Complete with new DMF3 Disk Controller - Save over \$1,000.00 - ONLY \$2475
Don't get caught without that **BACKUP** for all your data.
DON'T MISS THIS ONE! - Ask for Don, Tom or Larry, Days.

A/C 800 338-6800

LIKE NEW - ADDS-VIEWPOINT CRT TERMINALS - DEMO UNITS, THESE ARE LIKE NEW - WARRANTY NEVER REGISTERED - SAVE - ONLY \$478.00 EACH - ONLY 2 LEFT. CALL ASK FOR DON, TOM or LARRY

A/C 800 338-6800

GIMIX II CPU+, 2 Mhz, with FLEX and OS-9 - Level 1, Ver. 1.2, latest as of 9/1/83. This CPU board is like new, we went to GIMIX III system. Set up and ready to run. With FLEX and OS-9. Have both on your system and just call FLEX or OS-9. Includes software and documentation. Also Time of day clock, FLEX OAT, OS-9 FPLA. Set up to require GIMIX #68 DMA disk controller. Original cost over \$900.00. Special only 729.95, plus shipping (3.50). Ask for Don, Tom or Bob - 1 615 842-4601. **This will not last long!**

TELETYPE Model 43 PRINTER - with serial (RS232) Interface, and full ASCII keyboard. LIKE NEW - New cost \$1295.00 - ONLY \$759.00 ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

FOR SALE: SWTPC 6809 computer system configured for extended addressing. 64K RAM plus 16K RAM-DISK. Three disk drives (1-SA400, 2-SA455's), serial, parallel, timer, MPN calculator and EPROM programmer boards. Heath H-19 terminal. Software: FLEX09, ASMB, EDIT, Text Processor, Extended BASIC, BASIC Precompiler, Sort-Merge and more. \$1200 or best offer. 302-335-4758 after 5PM Peter Crossman

COMPILER EVALUATION SERVICES By: Ron Anderson

The S.E. MEDIA Division of Computer
Publishing Inc.
is offering the following **SUBSCRIBER SERVICE:**

COMPILER COMPARISON AND EVALUATION REPORT

Due to the constant and rapid updating and enhancement of numerous compilers, and the different utility, appeal, speed, level of communication, memory usage, etc., of different compilers, the following services are now being offered with periodic updates.

This service, with updates, will allow you who are wary or confused by the various claims of compiler vendors, an opportunity to review comparisons, comments, benchmarks, etc., concerning the many different compilers on the market, for the 6809 microcomputer. Thus the savings could far offset the small cost of this service.

Many have purchased compilers and then discovered that the particular compiler purchased either is not the most efficient for their purposes or does not contain features necessary for their application. Thus the added expense of purchasing additional compiler(s) or not being able to fully utilize the advantages of high level language compilers becomes too expensive.

The following **COMPILERS** are reviewed initially, more will be reviewed, compared and benchmarked as they become available to the author:

PASCAL "C" GSPL WHIMISCAL PL/9

Initial Subscription - \$39.95
(Includes 1 year updates)
Updates for 1 year - \$14.50

S.E. MEDIA - CPI
5900 Cassandra Smith, POB 794
Hixson, TN 37343
615 842-4601

GIMIX SYSTEM: 6800 CPU, 32K, Micropolis Drives, Serial and Parallel I/O, Windrush Eprom Programmer, SS30 Prototype Cards, Card Extenders, Humbug Monitor, Flex, Extended Basic, Debug Package and all manuals. As new. A \$4280. Value for \$2749.
Bob Dean Work: 604-721-7188, Home: 604-598-4113.

SWTPC 6800 W/24K, MP-C, MP-S, MP-L, MP-LA, Swatbug, CT-1024 Term, AC-30 W/1200 Baud, Expander Parallel Printer, Setchell-Carlson 15" Monitor, all cables and documentation including Edit/Assem, Basic, Disassembler. You pay shipping. Works good. Package deal only \$250.00 for all.
Pete Nissen, 7825 La Trec Dr, Jacksonville, FL 32221, (904)781-0972 anytime.

Wanted used 6800 cassette software JPC fast cassette, SWTPC Assembler, Basic, Word Processing, games, etc.
John Florino, 518 85th St, Brooklyn NY 11209.

MOONLIGHTERS 6800/01. We need experienced programmers on a continuing basis in Manchester, New Hampshire area. Use our development systems or yours.
Contact Karl Ritzinger 603-434-2300 (days) 603-669-4472 (evenings).

Glimix Mainframe, 6809 CPU "plus" DMA 5/8 controller, 80x24 video pcb, 2x32K Mem pcb, 2 Siemens 5" Drives, 1 Sanyo Dm 5112 CK Video Mon., RCA-Keyb, all boards Glimix. Price \$2000.00 or best.
Thomas Video pcb \$100, Glimix 8 parr. port \$110, Glimix 8 Serial port \$180, Glimix TT Rec. pcb \$150, Glimix 6800 cpu ++ \$100, Thomas Modem \$150, 2 8" SS DD Siemens Drives, In Vista V-1000 Cab. \$500, Teletype 43 RS-232 Int \$500 Sanyo Video Mon. Dm5112CX \$125.
Robert BaalJ 201-662-1826.

IDS 560 Paper Tiger \$600, Dual 8" DS/DD MPI Disk System including case, PS, and cable \$600, DS68 Graphics Board \$60.
Call Joe at (617)587-8241 after 7PM EST.

MUST SELL: All my SS50 cards. Enough for complete system(s). Call, the price is right.
David Pitts (203)528-8088 evenings.

HELP

Dear Don:

Please publish this shout for help.

I want to get in touch with anyone who can sell me one or more of listed software items.

1. Percom Super Basic + Manual
2. Percom Index + Manual
3. Percom General Ledger + Manual
4. Percom Mailing List + Manual
5. Percom Finder + Manual
6. List of Percom DFM/EPROM + Manual

Please drop me a line and tell me your name, address, and price per item.

My system is a 6800/SWTBUG/SS-50/SS-30 with Percom LFD 400.

Lennart Billgren
Soderleden 5
S 582 57 Linköping Sweden

Dear Mr. Williams:

I am really convinced that your Journal is the best source of information for 68XX(X) products and applications.

I would like to know if you have some information about these following topics;

- 1) Does it exist some formal 6809/68000 User Group in order to join efforts and save time, in the USA? I mean, if we could receive regularly 'standard-common' application routines (Assembler) on math, statistics, control, Op. Systems techniques, etc?

- 2) Has some of your subscribers written a Fast Fourier Transform routine for 6809/6800 (or Z80)? I would like to contact him, to interchange applications ideas.

- 3) Has some of your subscribers worked with the Extended Addressing for the SSB (6809 SCB-69, SS50C) system? I will appreciate him publishing a small sample routine accessing and running a program (assembler) on any address greater than 64K.

I would like that your magazine include gradually more 68000 information and application programs, I hope not bothering you with these questions.

Sincerely yours,
Prof Geza Holzhaker
Universidad de los Andes
Apdo Correos 454 Merida
Edo Merida Venezuela

DYNAMITE+™

"THE CODE BUSTER"

disassembles any 6809 or 6800 machine code program into beautiful source

- Learn to program like the experts!
- Adapt existing programs to your needs!
- Convert your 6800 programs to 6809!
- Automatic LABEL generation.
- Allows specifying FCB's, FCC's, FDB's, etc.
- Constants input from DISK or CONSOLE.
- Automatically uses system variable NAMES.
- Output to console, printer, or disk file.
- Available for all popular 6809 operating systems.

FLEX™ \$100 per copy; specify 5" or 8" diskette.

OS-9™ \$150 per copy; specify 5" or 8" diskette.

uniFLEX™ \$300 per copy; 8" diskette only.

For a free sample disassembly that'll convince you DYNAMITE+ is the world's best disassembler, send us your name, address, and the name of your operating system.

Order your DYNAMITE+ today!

See your local DYNAMITE+ dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Orders outside North America add \$5 per copy. Please specify diskette size for FLEX or OS-9 versions.

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020



uniFLEX software prices include maintenance for the first year.

DYNAMITE+ is a trademark of Computer Systems Center.

FLEX and uniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.
Dealer inquiries welcome.

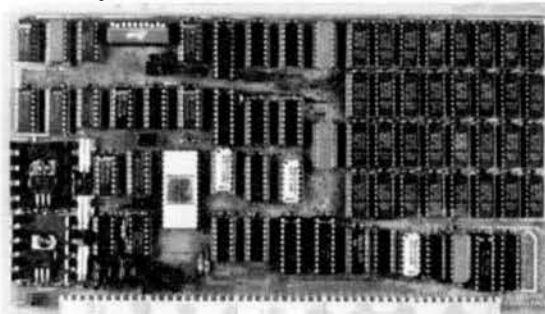
COMPUTER EXCELLENCE, INC.

IS PROUD TO PRESENT A SUPERB

256K MEMORY SYSTEM

FOR THE SS-50 OR SS-50c BUS.

Now any 6800 or 6809 system can have up to a MEGABYTE using our sophisticated new 256K memory system.



Runs full speed: 2MHz, no MRDY, no interrupts and still executes a refresh cycle EVERY system 'E' clock. Works with any processor card, with or without 'DATA's. High-speed DATA on board, standard. Works with any SS-50 bus, with or without extended addressing lines. No cables or bus modifications. Board is QUIET: over 8 uF of ceramic decoupling capacitance. All gold bus connectors, fully socketed, solder masked. Does not use the double height sockets. Sockets are selective gold plated for reliability. Assembled, tested, ready to go. Ready to work with the 256K Q-RAMS when they become available. Use all the memory, including \$E000-\$F000; no useless banks. Memory management scheme compatible with SWTPC and GIMIX. Virtual disk drivers, memory diagnostics supplied with board. Software is on 5 inch, SS, SD, 35 track disk for FLEX. Provides up to 3,750 sectors of virtual disk storage at RAM speed. Requires less than 1.2 Amps from your systems +8 volt bus. Many configuration options.

And it's ready NOW! Shipped from stock!

Introductory price: **\$749.00**

Including software, users and maintenance manuals or send \$15.00 for users manual, applicable toward purchase price. COD, Cashier's check, personal checks must clear before shipment. Florida residents add 5% sales tax.

4834 N.E. 12 AVE., FT. LAUDERDALE, FL 33334 (305) 752-8321

*FLEX is a trademark of Technical Systems Consultants

WORD PROCESSING SOFTWARE

SPELL 'N FIX finds and fixes your spelling and typographical errors. Cassette or disk versions cost just \$69.29 with a 200,000 word dictionary. FLEX version \$178.58. 75,000 word optional dictionary costs \$50 additional. ALL-IN-ONE editor word processor mailing list program costs \$50 (STAR-DOS version) or \$75 (FLEX version).

DISK OPERATING SYSTEM

STAR-DOS provides the power of a big DOS with the simplicity of standard R/S disk format. \$49.90 for 16K/64K systems.

SYSTEMS SOFTWARE

HUMBUG is the famous 6809 monitor/debugger adapted to the CoCo. \$39.95 for 16K or 32K disk or tape systems. \$59.95 for 64K systems using STAR-DOS or FLEX. \$29.95 for the MC-10. REMOTERM lets a terminal or modem control the CoCo or MC-10 for \$19.95. Disk or cassette. COMMTERM communications terminal program for the MC-10 costs just \$19.95. NEWTALK reads out memory contents in words through the TV speaker. \$20. Disk or cassette.

HOME FINANCE

CHECK 'N TAX lets you check on your bank. Not just a checkbook balancing program, but a help at tax time too. \$50, disk only.

EDUCATION

Numerical Methods is a college level course on computer mathematics. \$75, disk or cassette.

GAMES

SHRINK is our version of Eliza for \$15. Disk or cassette. THREE-D teaches spatial relationships through three-dimensional tic tac toe. \$25, disk or cassette.

STAR-KITS

P.O. BOX 209 - M
MT. KISCO, N.Y. 10549
(914) 241-0287

GOOD NEWS!



C for the 6809 WAS NEVER BETTER!

INTROL-C/6809, Version 1.5

Introl's highly acclaimed 6809 C compilers and cross-compilers are now more powerful than ever!

We've incorporated a totally new 6809 Relocating Assembler, Linker and Loader. Initializer support has been added, leaving only bitfield-type structure members and doubles lacking from a 100% full K&R implementation. The Runtime Library has been expanded and the Library Manager is even more versatile and convenient to use. Best of all, compiled code is just as compact and fast-executing as ever - and even a bit more so! A compatible macro assembler, as well as source for the full Runtime Library, are available as extra-cost options.

Resident Introl-C/6809 compilers running under Flex or OS9 are priced from \$375; Uniflex, from \$425.

Cross-compilers for PDP-11/Unix hosts are priced from \$1500.

Trademarks:

Introl-C, Introl Corporation;

Flex and Uniflex, Technical Systems Consultants;

OS9, Microware Systems;

PDP-11, Digital Equipment Corp.;

Unix, Bell Labs.

For further information, please call or write.

INTROL
CORPORATION

647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

Software

For Ordering Call TOLL FREE 1-800-338-6800 **FLEX™ OS-9™ Color Computer**

— DISASSEMBLERS —

Super Sleuth — Powerful, INTERACTIVE, Disassembler; provides Hex/ASCII Screen Dump of Memory Blocks or Disk Files for easy Examine/Change, Area Definitions; X-REF Generator of Source Code; change Address Labels to Names; etc. Disassembles 6800/1/2/3/5/8/9 and 6502 Code.

Color FLEX Obj. Only \$50.00, w/source \$99.00; FLEX \$99.00; UniFLEX \$100.00; OS-9 \$101.00

DYNAMITE + — An "easy to use" 6809 Disassembler for use w/ Disk Files, (will also disassemble 6800 Code). Develop a "Control File" of Area Definitions during successive disassemblies; X-Ref Source Files; replace Hex locations with Label Names; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX3, Color Computer FLEX.

FLEX and Color FLEX \$100.00 OS-9 \$150.00 UniFLEX \$300.00

— ASSEMBLERS —

TSC Macro Assembler — the FLEX STANDARD Assembler

FLEX and Color FLEX \$50.00

TSC Reloc. Asmb./Link. Load. — Need for many of the C and Pascal Comp. FLEX & Color FLEX \$150.00

RRMAC — Relocating, Recursive-Macro Assembler and Linking Loader for the 6809. Use either standard Motorola Format or Special Ed Smith Format. Supports Recursive Macros, Conditional Assembly, etc. Opt. X-Ref Listing; Includes a Small Line-oriented Editor as part of the Assembler. Greatly improved operating manual. FLEX and Color FLEX 6809 RRMAC w/Link and Editor \$150.00 w/Source, add \$50.00

OmegaSoft Relocatable Assembler and Linking Loader — 2-pass Reloc. Asmb; 2-pass Link. Load. Supports 6 Char. Labels, System Calls (SWI[x] FCB x), Expressions with Arith., Logic, and Shifts, etc.

#RALLI FLEX and Color FLEX \$125.00 (one year Maint. \$50.00)

MACE — (by Graham Trott) from WINDRUSH MICRO SYSTEMS. A combined Editor/Assembler designed to allow the Programmer to Enter, Edit, and Assemble Programs with a minimum of effort, w/o leaving the Program. **XPACE** is a Cross Assembler for the MC6800/1/3 and Hitachi HD6301 (CMOS 6801) with the same functions and features as MACE.

FLEX and Color FLEX - \$98.00

— A UniFLEX "basic" OS-Compiler —

DUB — Re-Create a Source Listing from UniFLEX Compiled BASIC Programs. Easy to Use; works w/ ALL Versions of UniFLEX basic; Output to Disk or Term. Time TESTED and PROVEN; SOLID! UniFLEX \$219.95

— COMPILERS —

PL/9 — (by Graham Trott) from WINDRUSH MICRO SYSTEMS. A "Structured" Assembly Language Editor/Compiler/Debugger, all in ONE PACKAGE; provides a totally INTERACTIVE Program Development Cycle. The Compiler supports large Symbol Names, Variable Types, Pointers, Control Structures, Stack, A-, B-, and D-Register manipulation, etc. The Source oriented Trace/Debugger provides Single Stepping, Breakpointing, etc. An excellent Software Development Tool for utilizing the power of the 6809.

FLEX and Color FLEX - \$198.00

C — (By James McCosh) from WINDRUSH MICRO SYSTEMS. SUPER C Compiler for the FLEX Operating System. Needs the TSC Relocating Assembler/Linking Loader for those "full blown" System Packages.

FLEX and Color FLEX - 295.00

Introl 6809 "C" Compiler; generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz 280 when using a 2MHz 6809 System (Re. p 43, "68" Micro Journal, May '83). Floats, etc.

FLEX, Color FLEX, OS/9 \$375.00 UniFLEX \$425.00

— PASCAL —

TSC PASCAL — Native Code Compiler (UCSD Oriented).

FLEX and Color FLEX \$200.00

Lucidata PASCAL — P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.

FLEX and Color FLEX 5" \$190.00 FLEX 8" \$205.00

OmegaSoft PASCAL — For the PROFESSIONAL; ISO Based, Native Code Compiler. For Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt Processing, ROM-able, PIC, Re-Entrant Code, etc. POTENTIAL! Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

#PCS2 FLEX and Color FLEX \$425.00 (one year Maint. \$100)

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

Add \$2.00 Per Item For Shipping & Handling

For Ordering Call TOLL
FREE 1-800-338-6800



FREE DISKETTE With Each \$50 Purchase

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343

(615) 842-4601

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

Software

For Ordering Call TOLL FREE 1-800-338-6800 FLEX™ OS-9™ Color Computer

— A full, screen oriented, WORD PROCESSOR —

STYLOGRAPH 2.0 -- (now runs on the Data-Comp and FHL Color FLEX Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL Color FLEX STYLO \$195.00; FLEX and OS-9 STYLO \$295.00; UniFLEX STYLO \$395.00

Fast SPELLING CHECKER -- allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors.

FLEX, Color FLEX, OS/9 \$125.00 UniFLEX \$175.00

MAIL MERGE -- greatly extends the power and flexibility of STYLOGRAPH. Allows Multiple Text files to be printed out as one large document. Provides for merging information into the Text File during printing (such as different names and addresses), etc.

FLEX, Color FLEX, OS-9 \$145.00 UniFLEX \$195.00

INFOMAG Data Base Management System -- An XBASIC-based, Menu Driven, DBMS with "Built-In" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This Time Proven DBMS will become the "Work Horse" of your Software Stable.

FLEX and Color FLEX \$295.00 UniFLEX \$395.00

Accts Rec., Accts Payable & Gen Ledger -- A FULL Accounting Package that can be used together, or as separate packages; provides the IRS required Audit Tracking. (XBASIC, based on the "Osborne Business Programs.")

FLEX and Color FLEX \$295.00/PROG UniFLEX \$395.00/PROG

An Electronic Spread Sheet

OTHECALC -- THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, I have just finished setting up a Full Junior College Electronics Curriculum using OTHECALC). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is FAST. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc.

FLEX and SPECIAL Color FLEX (Both FHL and Data-Comp) \$200.00 UniFLEX \$395.00

Machine Language DATA BASE MANAGEMENT System

Westchester Applied Business Systems XIMS Data Management Systems. Possibly one of the most powerful DBMS's available, this machine language program is small enough to operate on a single sided 5" disk, yet provides the speed of M.L. and power limited only by the user's imagination. Supports Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. Easy-to-use English Language Command Structure.

XIMS -- FLEX and Color FLEX \$179.95

XIMS + -- FLEX and Color FLEX \$250.00

UNIVERSAL DATA RESEARCH INC. -- Note: ALL Accounting and DBM Progs. Require FLEX and XBASIC. These are Time Tested programs from an old, established, software house; for Color FLEX Systems

Data Base Manager Part 1 - \$49.95; Data Base Manager Part 2 - \$49.95

Church Contributions - \$49.95 Single Entry Gen Ledger - \$49.95 Balanced Billing System - \$49.95

Integrated Software for Color FLEX

A/C \$99.95

A/P \$99.95

Gen Ledger \$189.00

Inventory 2 \$69.00

Payroll \$99.95

FLEX and UniFLEX -- Note: Requires XBASIC (FLEX) or basic (UniFLEX)

A/C - FLEX \$295

UniFLEX \$395

A/R - FLEX \$295

UniFLEX \$395

Gen Ledger - FLEX \$295

UniFLEX \$395

Inventory 2 - FLEX \$295

UniFLEX \$395

Payroll - FLEX \$295

UniFLEX \$395

DBM - FLEX \$350

UniFLEX \$450

Please specify 5 or 8 inch disk when ordering all software!

Computer Systems Consultants FLEX XBASIC Programs

FULL SCREEN FORMS DISPLAY

FLEX and Color FLEX \$50.00

UniFLEX \$75.00

FULL SCREEN MAILING LIST

FLEX and Color FLEX \$100.00

UniFLEX \$110.00

FULL SCREEN INVENTORY/REP

FLEX and Color FLEX \$100.00

UniFLEX \$150.00

TABULA RASA SPREADSHEET

FLEX and Color FLEX \$100.00

UniFLEX \$200.00

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

Add \$2.00 Per Item For Shipping & Handling

For Ordering Call TOLL
FREE 1-800-338-6800



FREE DISKETTE With Each \$50 Purchase

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343

(615) 842-4601

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

Software

For Ordering Call TOLL FREE 1-800-338-6800 **FLEX™ OS-9™ Color Computer**

SPELLB "Computer Dictionary" — OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are inputting Text with your favorite Editor or Word Processor. **SPELLB** is more than "another Spelling Checker"; it allows you to look up a word from within your Editor or Word Processor so that you ~~KNOW~~ it is right WHEN YOU TYPE IT IN with the **SPB.CMD** Utility (which operates in the **FLEX** Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. **SPELLB** first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. **SPELLB** also allows the use of Small Disk Storage systems.

FLEX and Color FLEX \$129.95

JUST — a Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any **FLEX** Users Software Library. **JUST** is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

- Output the "Formatted" Text to the Display for format analysis and change.
- Output the "Formatted" Text to a Text File for use with the supplied **PPRINT.CMD** for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
- User Configurable** for adapting to other Printers (comes set up for Epson MX-80 with Graftax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
- Automatic compensation for a "Double Width" printed line.
- Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
- Use with ANY Editor.
- Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

FLEX and Color FLEX \$49.95

SPECIAL! SPECIAL! SPECIAL!

Star-Kits excellent **SPELL'N FLEX Dictionary** and **WRITE 'N SPELL** Word Look Up Program IN ONE PACKAGE;

FLEX and Color FLEX Systems — BOTH for ONLY \$150.00

When these are gone; the price goes UP!! WAY UP!! ORDER NOW!!

Also, call for "More Info" on both the **FLEX** Based and Color Computer Based **STAR-Kits** Products; including the **HUMBUG** Monitor, **Check 'N Tax** Program, **REMOTERM** Color Computer External Terminal Program, etc.

PASCAL UTILITIES — Requires UCIDATA Pascal ver 3.

XREF — produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE — allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

PROFILER — produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

FLEX and Color FLEX — Each program \$25.00

COPYCAT -- (Pascal NOT required) Allows reading TSC Mini-FLEX, SSB 00668, and Digital Research CP/M Disks while operating under **FLEX 1.0**, **FLEX 2.0**, or **FLEX 9.0** with 6800 or 6809 Systems. **COPYCAT** will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in Modular Source Code to make it easier to solve unusual problems.

FLEX and Color FLEX 5" \$50.00 FLEX 8" \$65.00

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microvare

Add \$2.00 Per Item For Shipping & Handling

For Ordering Call TOLL
FREE 1-800-338-6800

SOUTH EAST MEDIA



5900 Cassandra Smith Rd., Hixson, TN 37343

FREE DISKETTE With Each \$50 Purchase

(615) 842-4601

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

Software

For Ordering Call TOLL FREE 1-800-338-6800 **FLEX™ OS-9™ Color Computer**

O-F — OS/9 to FLEX - FLEX to OS/9 —

Finally, the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ from, and WRITE to, a Single Sided 5" or 8" FLEX diskette from OS-9 with O-F. O-F is a new and unique program, written in BASIC99 (with Source), that performs the following functions;

REFORMAT: A BASIC99 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX.

FLEX: A BASIC99 Program that does the actual read or write function to the special O-F Transfer Disk, all selectable from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

FLEX and Color FLEX \$79.95

COPYMULT.CMD — Copy LARGE Disks to several smaller disks —

The following FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by COPYMULT, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included. ALL 4 Programs \$99.50 (8" or 5")

CHESS 6809

Requires FLEX and DISPLAYS On Any Type Terminal

Features:

- *Two display boards. *Change skill level. *Swap side. *Point scoring system.
- *Four levels of play. *Solve Checkmate problems in 1-2-3-4 moves.
- *Make move and swap sides. *Play white or black.

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

FLEX and Color FLEX \$79.95

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an X BASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

FLEX - \$59.95

UnifLEX - \$89.95

XDMA — A COMMUNICATION Package for the UnifLEX Operating System —

Allows UnifLEX Based Systems to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UnifLEX Systems, etc.

- Verifies Transmission integrity using checksum or CRC
- Automatically Re-Transmits bad blocks
- Transmits data in 128 byte blocks

UnifLEX \$299.99

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Marware

Add \$2.00 Per Item For Shipping & Handling

For Ordering Call TOLL
FREE 1-800-338-6800



FREE DISKETTE With Each \$50 Purchase

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343

(615) 842-4601

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

Software

For Ordering Call TOLL FREE 1-800-338-6800 **FLEX™ OS-9™ Color Computer**

AT LAST!! A FULL BLOWN DISASSEMBLER FOR THE COLOR COMPUTER

Computer Systems Consultants **SUPER SLEUTH** is a "Time Tested", reliable, **PROVEN** Disassembler that has gained acceptance through out the **FLEX** Community as an extremely **POWERFUL**, **INTERACTIVE**, Software Tool. Now, this powerful Disassembler has been converted to run on a **Standard 32K Color Computer** or **TDP-100 System with a Disk System**. The **CoCo SLEUTH** Software Package consists of 3 Programs; **SLEUTH** (the Disassembler), **CHGNAM** (used to globally Change Labels to a meaningful Name), and **XREF** (a Cross Reference Generator for Source Code Files). **CoCo SLEUTH** will disassemble Disk Files of 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Object Code if you can get it on a Color Computer Disk. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.) **Color Computer Disk - Object Code Only \$49.00**

FORTH Programming Language

Stearns Electronics FORTH -- Intrigued by **Forth777**? Here is a **Forth** package tailored to the **Color Computer**! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's **speed is unparalleled**. A full **Semigraphic-8 Editor** is provided, along with "goodies" like **Graphics** and **Sound Commands**, **Printer Commands**, **Auto-Repeat** and **Control Keys**, etc. If you are interested in **Learning Forth**, a **Trace Feature** is provided which is invaluable. If you are a **FORTH Pro**, this package provides **CPU carry flag accessibility**, **Fast Task Multiplexing**, **Clean Interrupt Handling**, etc. (Or; you won't "out grow" the **Basic capabilities** of this Implementation). Combine this package with **Leo Brodie's EXCELLENT Book "Starting FORTH"**, and you will be a **FORTH Expert** before you know it (and have a lot of fun doing it!).

Color Computer TAPE (w/ instructions for transferring to Disk) \$58.95

Color Computer GRAPHIC SCREEN PRINT Programs

Dumps any "PMODE" Screen to the Printer with the **BASIC USR** Function. Shift the Printout Left or Right or **Reverse Print** (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPP for Radio Shack LP-VII/VIII & DMP 100/200/400 Printers	\$7.95
GSPP2 for Epson w/ Grafrax and Grafrax + Printers	9.95
GSPP3 for Gemini 10 and 15 Printers	9.95
GSPP4 for the Prowriter Printers	9.95

DATE-O-BASE CALENDAR Program

A Menu Driven **EXTENDED BASIC** Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A **Graphic Calendar** shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR (Each Tape File will hold up to 400 Memos)	\$16.95
DISK DATE-P-BASE CALENDAR (4,000 Memos at 300/Month per Disk)	19.95

Interested in INTEREST (the Money Kind)?

An **EXTENDED BASIC** Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.

TAPE \$29.95

Data Base Management System

DISK DATA HANDLER 64K - EXTENDED BASIC w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first; 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely **POWERFUL TOOL**; instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System. **DISK \$54.95**

ACCOUNTING

DISK DOUBLE ENTRY - DISK EXTENDED BASIC w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. **DDE** allows up to 300 accounts and a Trial Balance of \$9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the journal transaction, which may include comments. Screen reports allow review of past transactions and current balances. **DISK \$44.95**

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

Add \$2.00 Per Item For Shipping & Handling

For Ordering Call TOLL
FREE 1-800-338-6800



FREE DISKETTE With Each \$50 Purchase

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343

(615) 842-4601

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

Software

For Ordering Call TOLL FREE 1-800-338-6800 **FLEX™ OS-9™** Color Computer

DYNASHARE — Multi-User, Multi-Tasking with **FLEX**

SouthEast Media is now shipping **DYNASHARE FROM STOCK** - the multi-user, multi-tasking capability of **DYNASHARE** allows **FLEX** users the advantages of more sophisticated and time saving computer usage without having to buy or learn a new Language or Operating System syntax. **DYNASHARE**, as its name implies, allows true "time-sharing" operation under the popular **FLEX** operating system, and also allows each user to run two simultaneous jobs (multi-tasking); even on single-user systems. For example, while in **EDIT**, you can list another file or examine a directory. Or, you might look up an item in a Data Base while a Sort is in progress! **DYNASHARE** also provides some fringe benefits that will be greatly appreciated by **FLEX** users, including type-ahead, command line editing, and instant response to "escape".

DYNASHARE is the painless method! Use your existing Flex computer by simply adding 64K of RAM for each user. Fact is, you still use **FLEX** just like you always have! **DYNASHARE** is not intended as competition to **UnifLEX**. It does not improve on the speed of **FLEX**, and does not offer password protection or other niceties of a full-blown multi-user system. What **DYNASHARE** does do is give **FLEX** users a low-cost way to use existing software in a multi-user, multi-tasking environment, so your existing **FLEX** versions of **BASIC**, **XBASIC**, editors, assemblers, disassemblers, sort/merge packages, word processors, compilers, **DYNACALC** spread-sheet package, and so on are still good.

NOTE -- The initial release of **DYNASHARE** is for **SWTPC 8/09** Computers, but versions will also be available for other popular extended-memory (up to 1024K) systems, such as **HELIX** and **GIMIX**. A minimum of 128K of RAM will be required with ALL versions. **DYNASHARE** requires 64K of RAM for each active task; thus a 256k system could allow foreground-background operation on two terminals, or foreground-only operation on four terminals.

AVAILABLE NOW from **SOUTHEAST MEDIA** — \$200.00

--- AUTHORS - PROGRAMMERS ----- QUALITY SOFTWARE NEEDED ---
FLEX - UNIFLEX - OS/9 - Color Computer

For the past several months, we at the **SouthEast Media Division of Computer Publishing, Inc. (CPI)**, the parent company of '68' **MICRO JOURNAL** and **COLOR MICRO JOURNAL**, have debated expanding our software distribution business. Many other magazines have been doing so for years (in fact, MOST were in the Software Distribution Business BEFORE they began to publish a Magazine). Presently there are many fine examples of software that has been developed by YOU, our readers, that will never see the "light of day" due to the cost of Advertising and TIME and cost involved in the production, distribution, and Customer SUPPORT of that software unless SOMEONE, with enough exposure and the willingness to continually advertise, runs with the ball.

Software is the "backbone" for the REAL utilization of any Computer System, and ours are no exception! This has been no simple decision. While we realize that there could be some conflict with some of our advertisers, we ALSO hear a LOUD and CONTINUOUS cry for HELP from our Readers. From day one, the foremost concern of '68' **MICRO JOURNAL** has been it's READERS! Therefore, our **SouthEast Media Division** will accept, for appraisal for possible Distribution, 6809 software; Games, Utilities, Software Development, Business Application Programs, etc.

In the past there has been too much software offered that was not quite ready. We will strive to eliminate that element. But, right up front, we tell you only that we will do our very best; nothing more. Also, we will strive to keep cost to a bare minimum, while securing for the author a fair return in royalty payments, promptly paid, and in customer support for his product.

Of course, we will expect, no -- **DEMAND**, that the author keep the product free of errors (bugs), and maintain it in a prompt and business like manner. Also we shall require that authors be willing to furnish 'source' for those programs that justify, by price and utility, inclusion of same. The lack of source code, properly commented, is a continual complaint we hear. Not all programs will be sold with source, but where necessary, we will insist that it be included.

In some instances the program may be small or short and not justify itself as a "single" sale product. In this event it will be combined with other like programs, and offered as a package. In that event, the royalties will be split between the various authors.

If you have software that you feel will qualify under this program, please contact one of the people below. Remember, if your software has any problems or "funnies" -- **GET IT STRAIGHT BEFORE YOU CONTACT US!!** Also get your source code in proper shape and well commented; there is too much 99% code already drifting around.

If your software is **READY** contact: **Bob Ray**, **Don Williams**, or **Tom Williams**

SouthEast Media is a division of **Computer Publishing, Inc. (CPI)**,
a family of 100% 68XX support facilities.

*FLEX is a trademark of Technical Systems Consultants

*OS9 is a trademark of Microware

For Ordering Call TOLL
FREE 1-800-338-6800

Add \$2.00 Per Item For Shipping & Handling

FREE DISKETTE With Each \$50 Purchase

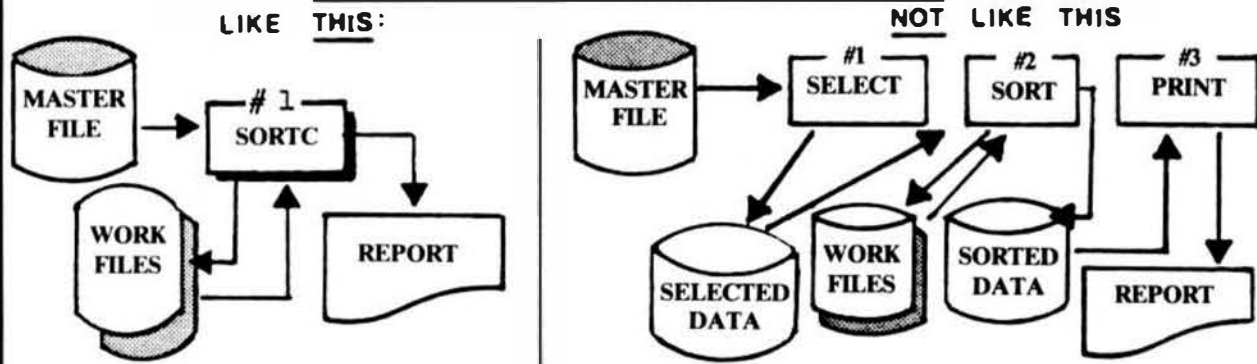
SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343

(615) 842-4601

SORTC** for OS9*

THE ONE AND ONLY



BOLDLY GOING WHERE NO SORT HAS GONE BEFORE

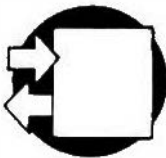
SORTC is a high speed, full-record compounding disk sort, which gives microcomputer users mainframe capabilities. It has been specifically designed to sort data efficiently while offering the user great flexibility in designing sort programs. It is written in BASIC09* for use under OS9.

COMPOUNDING FUNCTION

SORTC has the capability of summing user-specified numeric fields on equality of keys. This allows significant savings in memory, disk space, and program development time. A reduction in the number of disk accesses required when compared to other sorts is inherent in the design of SORTC.

DISK BASED

Specifically designed to sort large volumes of data, SORTC imposes no size restrictions on the amount of data to be sorted. It also places no limits on the number of sort keys which can be used or the order in which the keys are sorted. Furthermore, the sort procedure can be performed as many times as necessary within the same program. This feature allows the programmer to take advantage of any existing data bias, and possibly even reduce the size of the sort key.



JBM'S MIDDLEWARE

*OS9, BASIC09 are registered trademarks of Microware Corporation.

**Uses the same algorithm as JBM's SORTC for Digital Equipment Corp. RSTS Systems.

ADVANCED DESIGN

While most disk sorts are partially based upon the Fibonacci series, SORTC is not. SORTC is a generation ahead of the normal sorts based upon the "Fib series". Its unique algorithm is automatically optimized at run time for a reduction in workspace, reduced # of disk accesses and shorter run times. Designed to be as "crash proof" as possible, the sort procedure will not abort if it is accidentally asked to sort zero items.

EASY TO USE

It is not difficult to design a program which will use JBM's SORTC. Since SORTC is a subroutine, the user may write any procedure he or she wants to format the data for sorting and then to process the sorted data. The sorted data need not be written back to disk, but instead is immediately available. The sort code is automatically inserted into the source procedure by a simple Sort Generator.

ORDERING INFORMATION

SORTC, from JBM's MIDDLEWARE line of quality software, is available on either five and one-quarter or eight inch diskettes for a price of \$150.00. All of JBM's software packages come complete with comprehensive user's manuals.

For more information, or to place an order, contact:

DEPT. FSEA

The JBM Group, Inc.

332 West Church Road
King of Prussia, PA 19406

TEL: 215-337-3138

TWX: 510-660-3999

VISA and MASTERCHARGE accepted.

the JBM group

FLEXTM/UNIFLEXTM USERS...

TECHNICAL SYSTEMS
CONSULTANTS IS
EXPANDING ITS
TECHNICAL SUPPORT
STAFF. IF YOU MEET
THE FOLLOWING
QUALIFICATIONS:

- 1** EXPERIENCED
WITH FLEX
- 2** EXPERIENCED
WITH UNIFLEX
- 3** EXPERIENCED WITH OUR
SUPPORT SOFTWARE
- 4** HAVEDONE SOME
PROGRAMMING

SEND YOUR RESUME
AND SALARY

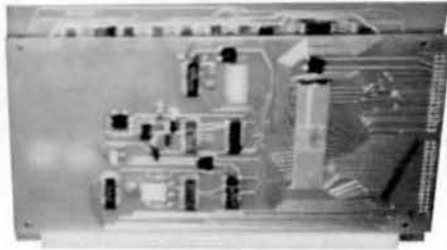
REQUIREMENTS TO:
TECHNICAL SYSTEMS
CONSULTANTS, INC.,
ATTENTION: TECHNICAL
EMPLOYMENT DEPT.,
111 PROVIDENCE
ROAD, CHAPEL
HILL, NC 27514.



FLEX AND UNIFLEX ARE TRADEMARKS OF TECHNICAL SYSTEMS CONSULTANTS, INC.

SUPER CPU

... only from
LSI



\$549.95

We're mighty proud of our new processor card. We're giving you the ability to go 68000 without major changes to your system. Our new CPU gives you these advanced features:

- Dynamic partitioning memory management unit with bound check register.
- On-board timer for multi-user/multi-tasking applications.
- On-board boot-strap EPROM and Monitor EPROM space.
- Vectored priority interrupt generator.
- On-board wait state generator.
- User selectable bus options that includes a new higher bandwidth bus mode.
- And many more...

S68K/08-CPU-ASSEM. & TESTED 549.95
S68K/08-CPU-KIT FORM 449.95
(KIT INCLUDES PROCESSOR, CRYSTAL,
SOCKETS AND CONNECTORS)
DISK CONTROLLER SUPPORTED DC3, DC4,
DMF2, SDC8.

ANNOUNCING THE LSI 68000 USERS GROUP

Join the Group by sending us your name and address. You will receive our monthly publication with free public domain user programs and software updates.

New members of the users group will receive a \$30.00 discount on THEIR FIRST LSI hardware purchase. Anyone that donates a program to the group will receive our current user group software on a formatted CP/M readable disk.

ANY ORDERS RECEIVED ON SDC8 OR THE 256K RAM CARD BEFORE 3-1-84 WILL RECEIVE A \$50 DISCOUNT

SDC8 CONTROLLED

A SS50 DMA disk controller for use with either 68000 bus modes or 6809 bus modes. Features a high reliability digital data separator. (No analog circuits to drift) and full 1 Megabyte addressing range.

S68K/08-SDC8-\$550.00

256K RAM CARD

Using the latest LSI technology this 256K RAM CARD makes a perfect addition to your S68K system. Uses MRDY for refresh arbitration.

S68K/08-256K ... \$750.00

CP/M-68K

DRI's famous CP/M made for the 68000. It includes a C compiler, relocatable assembler, linking loader, librarian and many utilities. It is source compatible with all other CP/M operating systems.

CP/M-68K ... \$350.00

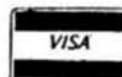
IDRIS-S68K

Named after the Persian deity of craftsman, is a full UNIX compatible time sharing operating system. It features a V7 compatible shell and is file compatible with UNIX V6. Includes 58 utilities. Requires a minimum 128K of RAM, 256K recommended for single users, 512K recommended for timesharing environment.

IDRIS S68K... \$650.00 WITH
PURCHASE OF SUPER CPU ONLY

LSI STANDS BEHIND ITS HARDWARE
1 YEAR LIMITED WARRANTY
ON ALL HARDWARE

• CP/M-68K is a registered trademark of Digital Research, Inc.
• UNIX is a registered trademark of Bell Labs
All prices and offers subject to change without notice.



N.Y. residents add sales tax

LSI Enterprises Ltd.

PO Box 1227
Woodhaven, NY 11421

(212) 423-5596

TEN MOST-ASKED QUESTIONS about **DYNACALC**TM

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. What is an electronic spread-sheet, anyway?

Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. Is DYNACALC Just for accountants, then?

Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. What will DYNACALC do for ME?

That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalcTM, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. Do I have to learn computer programming?

NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. Do I have to modify my system to use DYNACALC?

Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. Will DYNACALC read my existing data files?

You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPHTM word processor, RMSTM data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. How fast is DYNACALC?

Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. Is there a version of DYNACALC for MY system?

Probably. You need a 6809 computer (32k minimum) with FLEXTM, UniFLEXTM, or OS-9TM operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. How much does DYNACALC cost?

The FLEX versions are just \$200 per copy; UniFLEX version \$395; OS-9 version (works with LEVEL ONE or LEVEL TWO) \$250. Orders outside North America add \$7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. Where do I order DYNACALC?

See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020



UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of
Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

COMPARE

our EPROM PROGRAMMER with the field.

All data taken directly from manufacturer's current advertising. Software, interface, or personality modules may also be required at additional cost.

- Triple voltage EPROM
- Supplied in kit form

		A	B	C	D	E	F
INTERFACE	S30	PAR	PAR	SER	S30	SER	SER
INTELLIGENT	NO	NO	NO	YES	NO	YES	YES
PROGRAMS							
2704*	•		•		•	•	•
2508	•		•	•	•	•	•
2708*	•	•	•	•	•	•	•
2758	•	•	•	•	•	•	•
2516	•	•	•	•	•	•	•
2716	•	•	•	•	•	•	•
2718*	•	•	•	•	•	•	•
2532	•	•	•	•	•	•	•
2732	•	•	•	•	•	•	•
2732A	•	•	•	•	•	•	•
2584	•	•	•	•	•	•	•
2784	•	•	•	•	•	•	•
2528	•	•	•	•	•	•	•
27128	•	•	•	•	•	•	•
2816	•	•	•	•	•	•	•
68764	•	•	•	•	•	•	•
8748	•	•	•	•	•	•	•
8749	•	•	•	•	•	•	•
TOTAL	11	3	12	8	11	11	11
PRICE	\$125	\$45*	\$169	\$289	\$375	\$489	\$575

EPROM EPROM Programmer, \$125. Personality module for 2508, 2758, 2516, and 2716 included. Specify CPU, disk size, and operating system (TSC's FLEX or SBB's DOS) when ordering. Manual only, \$10; refundable with EPROM purchase.

UNITEK • P.O. Box 671 • Emporia, VA 23847

'68' MICRO JOURNAL

- ★ The only ALL 6800 Computer Magazine.
- ★ More 6800 material than all the others combined: **MAGAZINE COMPARISON**

(2 years)

Monthly Averages

KB	BYTE	6800 Articles	CC DOBB'S	TOTAL PAGES
7.8	6.4	2.7	2.2	19.1 ea. mo.

Average cost for all four each month: \$6.63
(Based on advertised 1-year subscription price)

'68' cost per month: \$2.04

That's Right! Much, Much More

for About

1/3 the Cost!

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge ☐ — VISA ☐

Card # _____ Exp. Date _____

For ☐ 1-Year ☐ 2 Years ☐ 3 Years

Enclosed: \$ _____

Name _____

Street _____

City _____ State _____ Zip _____

My Computer Is: _____

68 Micro Journal!
5800 Cassandra Smith Rd.
Hixson, TN 37343

SUBSCRIPTION RATES

USA

1 Year \$24.50, 2 Year \$42.50, 3 Year \$64.50

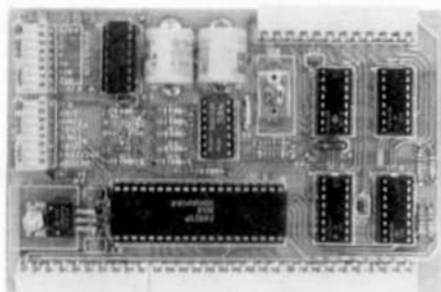
*FOREIGN SURFACE Add \$12.00 per Year to USA Price

*FOREIGN AIRMAIL Add \$36.00 per Year to USA Price

**CANADA & MEXICO Add \$5.50 per Year to USA Price
Cash (USA) or drawn on a USA Bank!!!



CALENDAR-CLOCK / TIMER / PARALLEL PORT



Calendar - Clock CLK68-1

- Reads date and time instantly by the computer in ms
- All 12-hr. increments displayed accurately
- On card battery (included) and charged circuit runs for months
- Set of 400, 1000, 10000, 100000 (12/24 hr)

Interval Timer

- For printer spooling, wait-loading, etc.
- Compatible with OS-9 and Flex 7.5
- Set of 400, 1000, 10000, 100000 (12/24 hr)
- Generates accurate intervals from 100 microseconds to 350 sec.

Parallel I/O Port

-- Fully buffered 8 bit parallel port

- 800 microsecond delay on output buffering the software on the board
- Compatible with parallel printer drivers to most versions of BASIC

Construction

-- Fully assembled, tested, and ready to go

Manual

-- Well documented - 36 pages

Prices + GST discount available

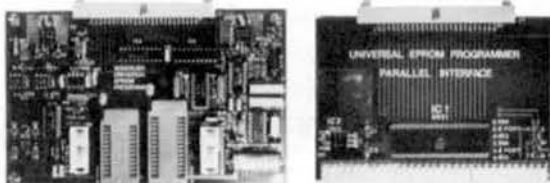
Assembled and tested	\$119.95	Kit	\$89.95
Goldplated bus cone	7.50	2 MHz option	2.50
Disk 5 or 8 in. SSS or Flex® OS-9 Available NOW			14.95

* OS-9 is a trademark of Microsoft Systems Corporation
* Flex is a trademark of Technical Systems Consultants, Inc.

ROBERTSON ELECTRONICS Phone (505) 294-0025
1003 Warm Sands Dr. SE NM residents add 4% tax
Albuquerque, NM 87123 Add \$3 Shipping & Handling

WINDRUSH MICRO SYSTEMS

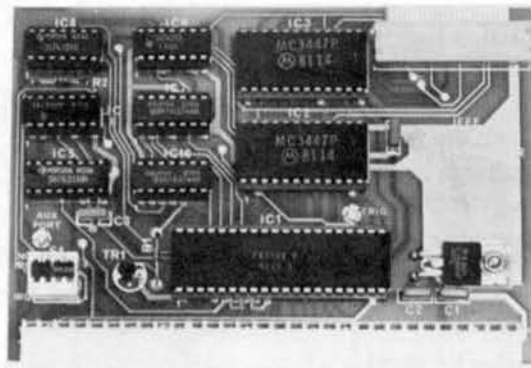
UNIVERSAL EPROM PROGRAMMER



- PROGRAMS and VERIFIES 2508, 2708, 2516, 2716, 2532, 2732A, 2564, and 2764 EPROMS. Minor hardware mods are required to program the INTEL 2712B.
- Tri-volt and Single Volt 2508/2708 and 2516/2716 devices are supported.
- ZIF sockets with mode selector switches eliminate 'personality modules'.
- Twin boards with five feet of twisted pair planar cable puts the programmer out on the bench where it belongs.
- SS-30 and EXORCISOR interfaces are available.
- Menu driven software provides the following facilities:
 - MOVE blocks of memory within the buffer.
 - READ an EPROM into the buffer.
 - VERIFY an EPROM against the buffer.
 - EXAMINE and change the contents of the buffer.
 - DUMP the contents of the buffer in HEX and ASCII.
 - FILL a selected area of the buffer with a specified character.
- Software available for all versions of SS0 DOS, FLEX 2, FLEX 9 and OS-9. Assembly language source files supplied on disk....enables customizing.
- Well documented users manual provides step-by-step adaptation and operating instructions.

AVAILABLE FROM GIMIX IN THE U.S.A.

IEEE-488



- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
 - Talker
 - Listener
 - System Controller
 - Serial Poll
 - Parallel Poll
 - Group trigger
 - Single or Dual Primary Address
 - Secondary Address
 - Talk only...Listen only
- Fully documented with a complete reprint of the SIOBAUD article on the IEEE bus.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. These drivers have been extensively tested and are GUARANTEED to work!
- Single SS-30 board (4, 8, or 16 addresses per port), fully socketed, gold plated bus connectors, and IEEE interface cable assembly.

PL/9 EDITOR/COMPILER/DE-BUGGER

- Friendly interactive environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the program a SOURCE line at a time. You also have direct access to any FLEX utility and your System Monitor.
- 250 page manual is organized as a tutorial with plenty of examples.
- Fast single pass compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files
- Signed and unsigned BYTES and INTEGERS, 32-bit floating point REALs.
- Vectors (single dimension arrays) and Pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (%), negation (~)
- Expression evaluators: (=), (>), (<), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/XOR).
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCB), (CCB) and (XREG).
- FULLY supports the MC6809 SWI, SWI2, SWI3, NMI, FIRQ, IRQ and RESET vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library function modules are supplied: ISSUES, BITIO, HARDIO, MEXIO, FLEXIO, SCIPACK, STRSUBS, and REALCON.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column. Need we say more?

WE STOCK THE FOLLOWING COMPANIES PRODUCTS:
GIMIX, SSB, FHE, MICROWARE, TSC, LUCIDATA, AND ALPORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporation, HDOS (tm) and EXORCISOR (tm) are trademarks of Motorola Incorporated.

MACE/XMACE

A co-resident EDITOR/ASSEMBLER for the 6809 written by Graham Trott which takes most of the pain out of assembly language program development:

- Friendly interactive environment where you have INSTANT access to the Editor, the Assembler, FLEX and your System Monitor.
- MACE can also produce ASMPPRC's for PL/9 with the assembly language source passed to the output file as comments.
- Includes XMACE a co-resident 6800/1/2/3/8 EDITOR/CROSS ASSEMBLER.

'C'

This is the FLEX version of the James McCosh 'C' compiler that is also available on UNIFLEX from SWTP and OS-9 from Microware:

- The FLEX implementation supports the full Kernighan and Ritchie 'C' specification except 'floats', 'doubles', and 'bit-fields'.
- Produces very efficient assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%
- Supports interleaved assembly language programs.
- The TSC relocating assembler/linking loader (SP09-17) is REQUIRED.

MACE	(6809 FLEX only).....	\$ 98.00
PL/9	(6809 FLEX only)...(a steal at this price!).....	\$198.00
'C'	(a 56K system and the TSC SP09-17 package is req'd)....	\$295.00
IEEE-488	with IEEE-488 cable assembly.....	\$298.00
SS-30	Universal EPROM Programmer w/one version of software...	\$375.00
EXORCISOR	Universal EPROM programmer w/one version of software...	\$395.00
SOFTWARE	Drivers for a 2nd, 3rd, or 4th operating system.....	\$ 25.00

ALL PRICES INCLUDE AIR MAIL POSTAGE

An SS-30C all CMOS 256K STATIC RAM board will be available SOON!

Write for details & pricing.

WORSTEAD LABORATORIES,
NORTH WALSHAM, NORFOLK,
ENGLAND. NR28 9SA.
TEL: (0692) 405189
TLX: 97360 SHARET G

FEATURES THE
POWERFUL, THIRD
GENERATION,
MOTOROLA 6809
PROCESSOR!

THE 6809 "UNIBOARD"TM SINGLE BOARD COMPUTER KIT

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL
AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY
FLOPPY DISK CONTROLLER!

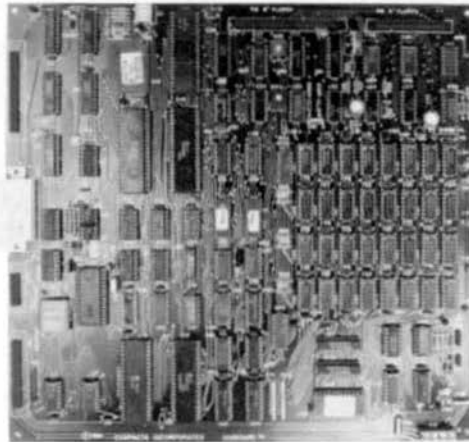
New!

BLANK PC BOARD

\$99⁹⁵

WITH PAL'S, AND
TWO EPROMS.

FOR 5-1/4 OR 8 INCH
SOURCE DISKETTE
ADD \$10.



\$399⁰⁰

COMPLETE KIT!
FULLY SOCKETED.

ALL OPTIONS ARE
STANDARD. NO
EXTRAS TO BUY!

THE COMPACTA UNIBOARDTM: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARDTM COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

FEATURES:

- * 64K RAM using 4116 RAMS.
- * 6809E Motorola CPU.
- * Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- * On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- * ASCII keyboard parallel input interface. (6522)
- * Serial I/O (6551) for RS232C or 20 MA loop.
- * Centronics compatible parallel printer interface. (6522)
- * Buss expansion interface with DMA channel. (6844)
- * Dual timer for real time clock application.
- * Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General FlexTM.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX TM from TSC	\$149
OS9 TM from Microware	\$199
Specify 5-1/4 or 8 Inch	

PC BOARD IS
DOUBLE SIDED, PLATED THRU
SOLDER MASKED, 11 x 11-1/2 IN.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY
LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

Digital Research Computers
(OF TEXAS)

P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 271-3538

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add \$4.00 shipping. USA AND CANADA ONLY

64K SS-50 STATIC RAM

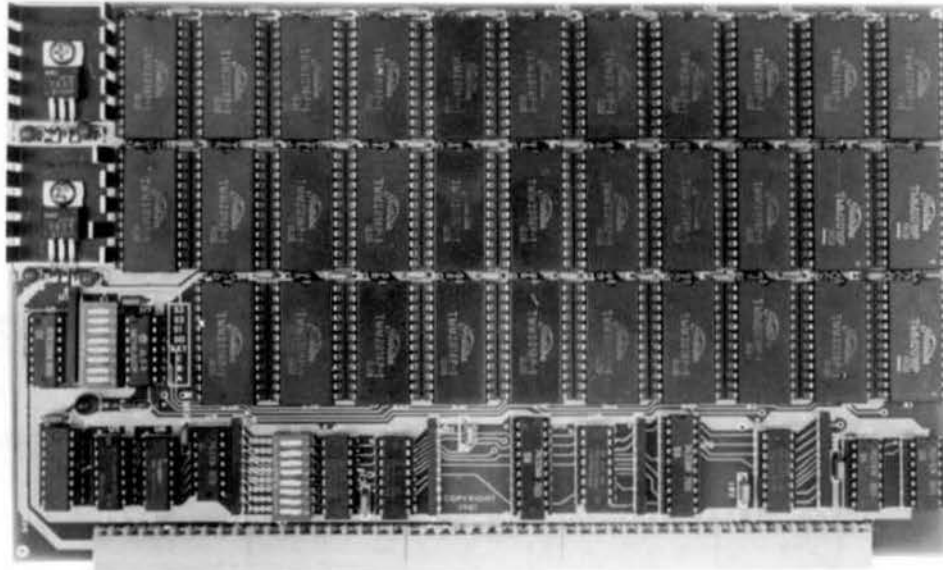
\$179⁰⁰
(48K KIT)

NEW!

NEW!

**LOW
POWER!**

**RAM
OR
EPROM!**



**BLANK PC BOARD
WITH DOCUMENTATION
\$52**

**SUPPORT ICs + CAPS - \$18.00
FULL SOCKET SET - \$15.00**

ASSEMBLED AND TESTED ADD \$50

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	\$219
64K	\$249

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

CLOSE OUT SPECIAL
WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR \$17.50 EA. THESE FORMERLY SOLD FOR \$50.

Digital Research Computers
(OF TEXAS)

P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add \$2.00 postage. We pay balance. Order under \$15 add 75c handling. No. C.O.D. We accept Visa and MasterCard. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50, add 85¢ for insurance.

Announcing . . . THE *SHELL* FOR FLEX 9™

We are pleased to announce the *SHELL*, a UNIX™ like shell that supports I/O redirection, pipes, macro substitution and programmable shell scripts! The shell will work with all your existing programs and utilities. Requires 56K of user ram, FLEX 9™ version 2.6 and above. The shell occupies the top 8K of user ram. An excellent tool for the 6809 community.

FLX/SHO9-8...8 inch version 90.00
FLX/SHO9-5...5.25 inch version ... 90.00
ONE YEAR MAINTENANCE 22.50

UNIX is a trademark of Bell Labs™.
FLEX 9™ is a trademark of Technical Systems Consultants Inc.

All prices subject to change without notice.

VISA

MC

COD

NY residents add sales tax.

LSI Enterprises Ltd.

PO Box 1227
Woodhaven, NY 11421
(212) 423-5596

F&D Associates
1210 Todd Road Color Computer & S50
New Plymouth, Ohio Boards & Accessories
45654 614 592 5721

Send for free Catalog
Visa ~ Master Charge ~ C.O.D.



RDC-18
COLOR COMPUTER
FLOPPY DISK
CONTROLLER BOARD

The RDC-18 uses 1793 FDC chip and SMC92168 single chip data separator and also has digital write precomp so no adjustments are required. Both edge connectors are BOLD plated. It accepts a 24 or 28 pin ROM or EPROM allowing up to 16k of on-board memory and is completely Radio Shack compatible if the RS Disk Extended Basic ROM is used. We have software to operate FLEX* using our FADBUG-C monitor. (CCFLXCON)

RDC-18AS	assembled/tested with case	\$169.95
RDC-18	bare board, doc. and 92168	\$54.50
CCFLXCON	diskette and instructions	\$39.95
DEBROM	Disk Ex Basic ROM(1.1)	\$29.95
BDCASE	case for board	\$ 7.50
1793	FDC CHIP	\$32.00
FADBUG-C	2716 EPROM monitor & manual	\$25.00

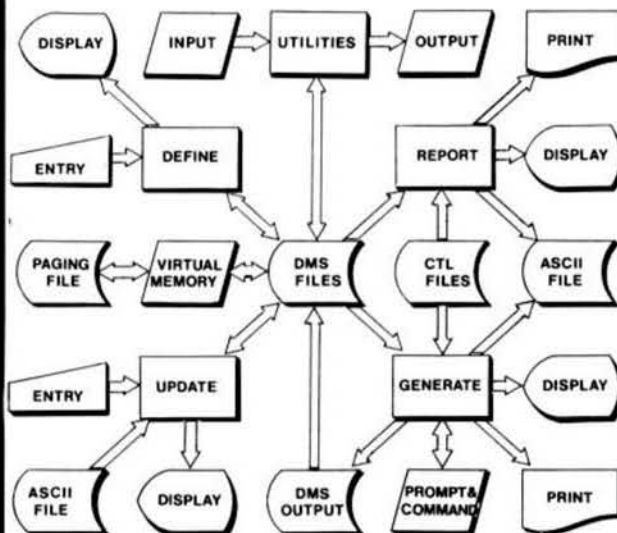
ASK ABOUT PACKAGE PRICES

add \$3 s/h. Oh res add 5 percent.

* Trademark of Technical Systems Consultants

XDMS

Data Management System



System Architecture

WESTCHESTER Applied Business Systems
Post Office Box 187
Briarcliff Manor, N.Y. 10510

XDMS Data Management System

The XDMS Data Management System is available in three levels. Each level includes the XDMS nucleus, VMOEN utility and System Documentation for level III. XDMS is one of the most powerful systems available for 6809 computers and may be used for a wide variety of applications. XDMS users are registered in our database to permit distribution of product announcements and validation of user upgrades and maintenance requests.

XDMS Level I

XDMS Level I consists of DEFINE, UPDATE and REPORT facilities. This level is intended as an "entry level" system, and permits entry and reporting of data on a "tabular" basis. The REPORT facility supports record and field selection, field merge, sorting, line calculations, column totals and report titling. Control is via a English-like language which is upward compatible with level II. XDMS Level I \$129.95

XDMS Level II

Level II adds to Level I the powerful GENERATE facility. This facility can be thought of as a general file processor which can produce reports, forms and form letters as well as file output which may be re-input to the facility. GENERATE may be used in complex processing applications and is controlled by a English-like command language which encompasses that used by Level I. XDMS Level II \$199.95

XDMS Level III

Level III includes all of level II plus a set of useful DMS Utilities. These utilities are designed to aid in the development and maintenance of user applications and permit modification of XDMS system parameters, input and output of XDMS files, display and modification of file format, graphic display of numerical data and other functions. Level III is intended for advanced XDMS users. XDMS level III \$269.95
XDMS System Documentation only \$10. credit toward purchase. . . \$ 24.95

XACC Accounting System

The XACC General Accounting System is designed for small business environments of up to 10,000 accounts and inventory items. The system integrates accounting functions and inventory plus general ledger, accounts receivable and payable functions normally sold separately in other systems. Features user defined accounts, products (or services), transactions, invoicing, etc. Easily configured to most environments. XACC General Accounting System (Requires XDMS, pref. Lv. III) . . \$299.95
XACC System Documentation only \$10. credit toward purchase. . . \$ 24.95

WESTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, N.Y. 10510

All software is written in macro/assembler and runs under 6809 FLEX O/S. Terms: Check, Money Order, Visa or Mastercard. Shipment first class. Add P&H \$2.50 (\$7.50 Foreign). NY Res add sales tax. Specify 5" or 8".

Sales: S. E. MEDIA, 1-800-330-6800, Consultation: 914-941-3552 (evening).

FLEX is a trademark of Technical Systems Consultants, Inc.

AMX

Real-Time Multitasking Executive

for
8080, Z80, 6809
and 8086

Gives your application a head start

Save time and money in the development of your product or system by using AMX, the software executive with proven, fault-free operation.

SIMPLE OPERATION

You divide complex control programs into a number of separate, more manageable programs, called tasks, each designed to do one job. This allows tasks to be written and tested separately and then combined to form a reliable, finished system.

AMX supervises the orderly execution of these tasks, assuring that the most important jobs always get done first. Tasks appear to be executing simultaneously. It's almost like having a separate CPU for each task!

HARDWARE INDEPENDENCE

AMX does not require a particular hardware configuration. You control your environment. You pick the I/O method. You decide the preferred interrupt service technique for your system. AMX will support you on the microprocessor of your choice.

AMX is fast, compact, and ROMable. The AMX nucleus, less than 1400 bytes in size, features multiple task priorities, intertask message passing with priority queuing, external event synchronization, and interval timing.

Support modules provide extended memory management, buffer control and resource allocation. Fast, re-entrant integer and floating point math libraries are also available.

AMX interfaces support programs written in C, PASCAL, PL/M, FORTRAN and assembler.

Access to CP/M® disk files in real time is possible using the AMX I/O Supervisor.

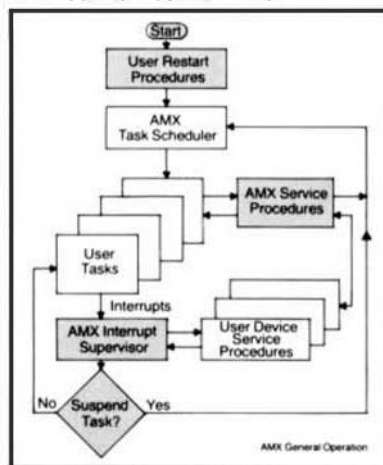
COMPLETE DOCUMENTATION

We deliver AMX source on diskette to permit AMX to be moved to the software development system of your choice. Our liberal license agreement permits binary (object) distribution without royalties.

HOW TO ORDER

A specification sheet and price list are available, free. Your check or money order for \$75 will purchase the AMX Reference Manual for immediate evaluation (specify 8080, Z80, 8086 or 6809 processor). Add \$25 for postage and handling outside USA and Canada. The standard 8080/Z80 AMX Multitasking Executive package, including source code, is **\$800**. Support modules and interfaces are available separately.

AMX is the choice of professionals the world over. Make it yours, today.



AMX is a trademark of KADAK Products Ltd.
CP/M is a trademark of Digital Research Corp.
Z80 is a trademark of Zilog Corp.



KADAK Products Ltd.



206-1847 W. Broadway Ave., Vancouver, B.C., Canada V6J 1Y5/Phone: (604) 734-2796 Telex: 04-55670

6809 Word Processing System

*stylograph*TM

STYLOGRAPH 2.0
The "User Friendly" word processing system. Fewer key strokes by the operator make it easier to learn.
OS9, FLEX \$295 UNIFLEX \$395
COLOR COMPUTER FLEX \$195

SPELLING CHECKER
Checks all words against an internal user-expandable dictionary of over 42,000 words.
OS9, FLEX \$145 UNIFLEX \$195

MAIL MERGE
Inserts names and addresses into form letters and mailing lists. Appends files at print out time. Handles files longer than memory.
OS9, FLEX \$125 UNIFLEX \$175

Inquire about our other software
• Business Programs - G/L, A/R, A/P
• Data Base Management System
• Assemblers
Also, Daisy Wheel Printers \$599.

Great Plains Computer Company Inc.
P.O. Box 916
Idaho Falls, Idaho 83401
(208) 529-3210

Flex and Uniflex are trademarks of Technical Systems Consultants, Inc.
OS9 is a trademark of Microware.

**AM-100™ ADVANCED WINCHESTER
SubSystems for FLEX™ and OS-9™
COMPLETE™ AM-100 starting as low as \$1899.-**



complete subsystems**

5MB FIXED -	STD. 5" -	\$1899.-
10MB FIXED -	STD. 5" -	\$2099.-
5MB FIXED -	MINI 5" -	\$2315.-
5MB REMOVABLE -	MINI 5" -	\$2395.-

** COMPLETE SUBSYSTEM = Drives(s) + Controller + Power Supply + Enclosure + All Cables + Host Interface + Software Drivers NO HIDDEN EXTRA !!

options

SS-50 HOST INTERFACE incl. SOFTWARE DRIVERS	\$325.-
SET of WINCHESTER SOFTWARE UTILITIES	\$49.-

shipping & handling 2.00 (U.S. Canada) 4.00 (overseas)

TERMS: - VISA - MONEY ORDER -

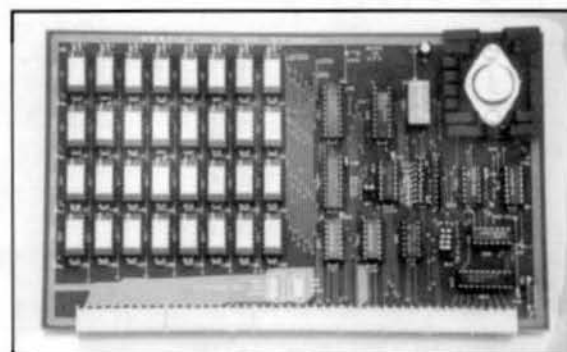
Tel.: (514) 737-8787



interfacing technologies corp.

P.O. Box 578, Snowdon, 4890 Bourret Ave. Montreal, Que., H3X 3T7

Trademarks: 1. i.t. Interfacing Technologies Corp. 2. Technical Systems Consultants Inc. 3. Microware Inc.



**QMM2 \$795
256K DYNAMIC MEMORY**

Works in SS50-C 6809 systems by SWTPC, SSB, GIMIX, and others. Runs at 1 or 2 Mhz with on board refresh. Comes assembled, tested, burned in and warranted for 1 year.

Professional quality board with socketed components and gold connectors.

Also available with 64K, 128K, or 192K.

Delivery: Stock—2 weeks.

Terms: Prepaid, C.O.D., VISA, MASTERCARD.

D.P. Johnson (503) 244-8152
7855 S.W. Cedarcrest St., Portland, OR 97223

OS-9 PROBLEMS?

Call: 215-337-3138

Telex: 510-660-3999

Write: The JBM Group, Inc.
332 West Church Road
King of Prussia, PA
19406

The OS-9*Solution Team

*Trademark of Microware and Motorola

68 MICRO JOURNAL PROGRAMS on DISK

Disk #1: FILESORT, MINICAT, MINICOPY, MINIFMS,
**LIFETIME, **POETRY, **FOODLIST, **DIET.
Disk #2: DISKEDIT w/ inst. & fixes, PRIME, **PMOD,
**SNOOPY, **FOOTBALL, **HEXPAMN, **LIFETIME.
Disk #3: CBUG09, SEC1, SEC2, FIND, TABLE2, INTEXT,
DISK-EXP, *DISKSAVE.
Disk #4: MAILING PROGRAM, *FINDDAT, *CHANGE,
*TESTDISK.
Disk #5: *DISKFIX 1, *DISKFIX 2, **LETTER,
**LOVESIGN, **BLACKJAK, **BOWLING.
Disk #6: **PURCHASE ORDER, INDEX (Disk file Indx).
Disk #7: Linking Loader & RLOAD, Harkness
Disk #8: CRTSET, Lanpher (May '82)
Disk #9: DATECOPY, DISKFIX9 (Aug '82)

NOTE: All are as published or received by 68
Micro Journal, some have fixes and patches.

This is a reader service only! No Warranty is
offer- or implied, they are as received and are
for reader convenience ONLY. Also 6800 and 6809
programs are mixed, as each is fairly simple
(mostly) to convert to the other.

PRICE: 8" Disk \$19.95 - 5" Disk \$17.95

68 MICRO JOURNAL
POB 794
Mixon, TN 37343
615-842-4600

* Indicates 6800, ** Indicates BASIC SWTPC or
TSC - 6809 no Indicator.

MASTER CARD - VISA accepted - Foreign add
sufficient postage surface or air!!

ANDERSON COMPUTER CONSULTANTS & Associates

Ron Anderson, respected author and columnist
for 68 MICRO JOURNAL announces the Anderson
Computer Consultants & Associates, a con-
sulting firm dealing primarily in 68XX(X)
software design. Our wide experience in
designing 6809 based control systems for
machine tools is now available on a
consultation basis.

Our experience includes programming
machine control functions, signal analysis,
multi-axis servo control (CNC) and general
software design and development. We have
extensive experience in instrumentation and
analysis of specialized software. We support
all popular languages pertaining to the 6809
and other 68XX(X) processors.

If you are a manufacturer of a control or
measuring package that you believe could
benefit from efficient software, write or call
Ron Anderson. The fact that any calculation
you can do with pencil and paper, can be done
much better with a microcomputer. We will be
happy to review your problem and offer a
modern, state-of-the-art microcomputer
solution. We can do the entire job or work
with your software or hardware engineers.

Anderson Computer Consultants & Associates
3540 Sturbridge Court
Ann Arbor, MI 48105

MACROPLEX Software

175 Fifth Avenue, Suite 3011, New York, NY 10010

Presenting... **THE CONDUIT** A new File Type for OS9

THE CONDUIT lets programs exchange data without using external files. Direct program-to-program data transfers on any path use OS9 I/O calls. No need to change device-independent programs. Two-way transfers are permitted and device-dependent code can be accommodated. Programs that formerly ran sequentially can now run concurrently to the limit of memory.

Utilities supplied with **THE CONDUIT** give immediate results when linked to your current programs via **CONDUIT** paths:

- Fast disk store-down speeds up assemblies and other I/O bound procedures
- "In-flight" terminal I/O redirection (e.g., set traps in DEBUG from a disk file)
- Basic OS9 source inclusions, enhanced listing
- "Null files" drain output, signal EOF on input

Free brochure available. Write, or call (212) 686-3036.

Macros alone (50 utilities)	\$ 8
CONDUIT file system + utilities	\$70
Both together	\$75
Package outside North America, add	\$ 5

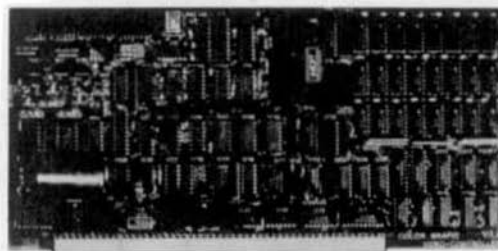
Specify 5" or 8" disk

Version for OS9 Level Two available soon.

OS9™ Basic OS9™ Microware Systems Corp. & Motorola, Inc.

Multi-GRAPH

Multimode-Color-Graphics-Video-Controller



for SS50C-Bus, the ultimate all in one graphics controller:

- 512x512 dot high resolution, 14 MHz dot clock • 192 kbytes on board screen memory for 2 independent pictures • programmable color-look-up-table for 8 of 4096 progr. colors
- b/w modes include 8 level of greyscale • X/Y screen-memory-addressing by 12 bit registers, so there are no wrap-around effects (!) • sym. X and Y resolution, lines have the same length in X- and Y-direction (!) • prog. hardw. ASCII char. generator with basic resolution of a 58 dots, programmable independent hor. and vert. 1...16 zoom, tilted characters in X and Y direction • hardw. vector generator with 4 types of lines, 1,500,000 dots/s • 3 powerful operation modes, simult. colored and b/w display, special b/w mode allows selection of any combination of memory-planes to display with no grey scaling (!)
- read-modify-write hardware for non-destructive overwriting of cross-hairs, cursor or other figures (!) • 8-bit grayscale, 16 or 20 bit addressing
- 1 or 2 MHz operation • indep. programmable hardware-blinking on all 3 planes • programmable write-protection and display sel. for all planes • built-in lightpen interface (on dot basis) • interrupt-generation on IRQ or FIQ from many sources • programmable comp. video input allows working with only one monitor for terminal and graphics display (in b/w mode) • universal RGB and comp. video outputs, CCR 625 lines, 50 Hz, interl. dependent only on monitor-sync, not on line frequency (!) nearly all quality RGB monitors are available for this format (!) • designed for easy software implementation, plotter-like programming • SS50C bus, high quality pc-board, soldermasked and silk screened non standard format: 5.8" x 12", but fits into most SWTPC and GIMIX systems (see picture).

Standard delivery is:
high quality pc-board, schematics, manual, EF 9385 data sheet, subrouting package in ASSEMBLER (8809), BASIC, C, PASCAL, FORTRAN77 for easy program inheritance.
Source code on FLEX09 compatible floppy.

- SS50C-Bus board with all features \$ 1670
- high speed lightpen with 50 nsec delay and switch \$ 335
- special adapters are available for APPLE and COMMODORE (include case with power supply) please ask for detailed information FLEX09 is trademark of SWTPC

Dipl. Ing. Jürgen Hnauft

SOFTWARE HARDWARE DIGITALELECTRONIC

O 6457 Mühldorf 1 (West Germany)
Birkmaring 1 - Tel. 0 61 81 / 4 56 43

SOFTWARE For THE HARDCORE

** FORTH PROGRAMMING TOOLS from the 68XX&X **
** FORTH specialists — get the best!! **

NOW AVAILABLE — A variety of rom and disk FORTH systems to run on and/or do TARGET COMPILATION for
6800, 6301/6801, 6809, 68000, 68080, 280

Write or call for information on a special system to fit your requirement.

Standard systems available for these hardware—

EPSON HX-20 rom system and target compiler
6809 rom systems for SS-50, EXORCISER, STD, ETC.
COLOR COMPUTER
6800/6809 FLEX or EXORCISER disk systems.
68000 rom based systems
68000 CP/M-68K disk systems, MODEL II/12/16

tFORTH is a refined version of FORTH Interest Group standard FORTH, faster than FIG-FORTH. FORTH is both a compiler and an interpreter. It executes orders of magnitudes faster than interpretive BASIC. MORE IMPORTANT, CODE DEVELOPMENT AND TESTING is much, much faster than compiled languages such as PASCAL and C. If Software DEVELOPMENT COSTS are an important concern for you, you need FORTH!

firmFORTH™ is for the programmer who needs to squeeze the most into roms. It is a professional programmer's tool for compact rommable code for controller applications.

- FORTH and firmFORTH are trademarks of Talbot Microsystems
- FLEX is a trademark of Technical Systems Consultants, Inc.
- CP/M-68K is trademark of Digital Research, Inc.

tFORTH™ from TALBOT MICROSYSTEMS NEW SYSTEMS FOR 6301/6801, 6809, and 68000

---> tFORTH SYSTEMS <---

For all FLEX systems: GIMIX, SWTP, SSB, or EXORCISER Specify 5 or 8 inch diskette, hardware type, and 6800 or 6809.

- ** tFORTH — extended fig FORTH (1 disk) \$100 (\$15)
with fig line editor.
- ** tFORTH+ — more! (3 5" or 2 8" disks) \$250 (\$25)
adds screen editor, assembler, extended data types, utilities, games, and debugging aids.
- ** TRS-80 COLORFORTH — available from The Micro Works
- ** firm FORTH — 6809 only. \$350 (\$10)
For target compilations to rommable code.
Automatically deletes unused code. Includes HOST system source and target nucleus source. No royalty on targets. Requires but does not include tFORTH+.
- ** FORTH PROGRAMMING AIDS — elaborate decompiler \$150
- ** tFORTH for HX-20, in 16K roms for expansion unit or replace BASIC \$170
- ** tFORTH/68K for CP/M-68K 8" disk system \$290
Makes Model 16 a super software development system.
- ** Nautilus Systems Cross Compiler
— Requires: tFORTH + HOST + at least one TARGET:
— HOST system code (6809 or 68000) \$200
— TARGET source code: 6800-\$200, 6301/6801-\$200
same plus HX-20 extensions— \$300
6809-\$300, 68080/280-\$200, 68000-\$350

Manuals available separately — price in ().
Add \$6 system for shipping, \$15 for foreign air.

TALBOT MICROSYSTEMS 1927 Curtis Ave., Redondo Beach, CA 90278 (213) 376 9941

FREE CATALOG

FH **FRANK
HOGG
LABORATORY**
THE REGENCY TOWER • SUITE 215 • 770 JAMES ST. • SYRACUSE, NY 13203
PHONE (315) 474-7856 • TELEX 646740

ARCADE 50

POWERFUL COLOR GRAPHICS

Uses the new TMS9918A Video Display processor. High resolution 256 x 192 pixel display with 15 colors. 16K Bytes of onboard RAM does not reduce user memory. 32 graphic images can be individually moved with simple X-Y commands for smooth animation. External Video input allows subtitling. NTSC composite video output.

SOUND EFFECTS AND MUSIC

- Three A 3-8910 Programmable Sound Generators
- Nine simultaneous voices
- Three independent noise sources
- On board stereo amplifier drives two 8 ohm speakers

ADDITIONAL I/O CAPABILITIES

- Eight analog inputs with 8 bit resolution
- Supports four joysticks with pushbutton switches
- Eight bit parallel I/O port
- Entire unit maps into 256 bytes of memory

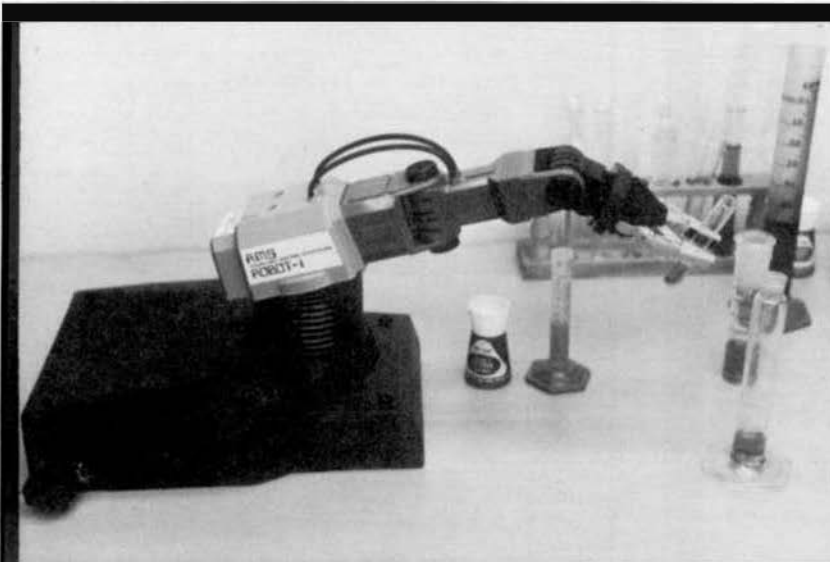
FBASIC

TERMINUS DESIGN INC. in conjunction with Microware Systems Corporation, is proud to announce FBASIC an enhancement of Microware's 6800/BASIC. Their fast compiled BASIC has been adapted for 6809 users with added video and sound features for ARCADE 50 users. FBASIC is a true compiler that produces optimized machine language modules which are ROMable and require no Run-Time package. FBASIC requires less memory overhead and runs hundreds of times faster than BASIC interpreters. It supports standard BASIC instruction including String functions. Disk I/O and fast integer arithmetic with multiple-precision capability. Graphics verbs and functions fully support the Arcade 50.

ARCADE 50 assembled and tested	\$325.00
Video and Audio connector set	15.00
4 Joystick connector set	15.00
2 Radio Shack Joysticks	24.00
Gold Molex connectors	12.00
A/BASIC for 6800	110.00
FBASIC for 6809	110.00
FBASIC (with ARCADE 50)	75.00
ARCADE 50 RGB	375.00
LABVIDEO (Motorola EXORbus)	375.00
NEW MV09 6809 Processor Board	225.00
256K Dynamic Memory Board	795.00
256K Dynamic Memory Board 1w/64K	395.00
64K Dynamic Memory Board	295.00

TERMINUS DESIGN INC
16 SCARBROUGH ROAD
ELLENWOOD, GA 30049

TERMINUS DESIGN, INC. (404) 474-4866



ROBOT-1

by


ANALOG MICRO SYSTEMS
5660 Valmont Road
Boulder, Co. 80301

Includes SS-30 card with power and 6 channel servo controller, Robot-1 with cables and software.
Order Robot-1s \$395. ea.

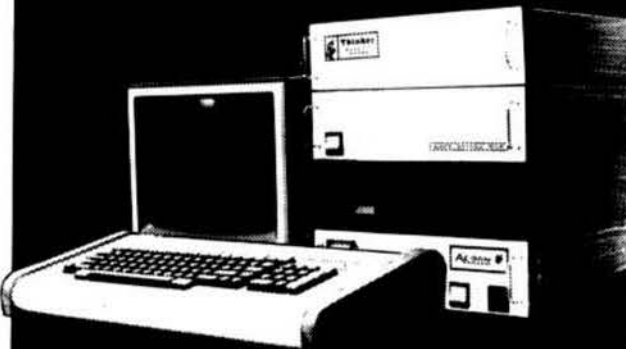
Interface for 4 joysticks and switches
Order ADS8 \$195. ea.

Free Catalog

Up to 8 robot arms with 8 SS-30 servo control cards may be programmed to operate interactively and simultaneously using the AMS multi robot arm software. All movement sequences may be saved on disc and retrieved later for repeating the multi interactive robot arm movement sequences. Source code is provided with all software to enhance the ease of research and experimentation in the field of robotics. Phone (303) 444-6809




ACORN
COMPUTER SYSTEMS 88-50C



MODULES - BARE CARDS - KITS - ASSEMBLED & TESTED			
Stackable Modules		KIT	A&T
20 amp POWER SUPPLY w/fan w/Disk protect relay		350.00	400.00
DISK CABINET w/rage. & cables less DRIVES		200.00	250.00
MOTHER BOARD, 8 88-50c, 8 88-30c NMI button		225.00	325.00



Item	Bare	KIT	A&T
IT3 - INTERRUPT TIMER 1, 10, 100 per sec.	19.95	29.95	39.95
PB4 - INTELLIGENT PORT BUFFER Single board comput.	39.95	114.95	139.95
DP1A - Dual PIA parallel port, 4 buffered I/Os	24.95	89.95	89.95
KADR - Extended Addressing BAUD gen. PIA port	29.95	89.95	89.95
MB8 - MOTHER BOARD 88-50c w/BAUD gen.	84.95	149.95	199.95
P168 - 168K PROM D16X 21, 2764 EPROMs	39.95	79.95	109.95
FD88 - Firmware development 2, 8K blocks	39.95	84.95	114.95
EMPR - 2764 PROM burner adapt. for 2716 BURNER	19.95	-----	-----
CHEART Keyboard w/Cabinet 96 key capacitive	249.95	-----	-----
TAXAN 12", 18 Mhz MONITOR GREEN AMBER	-----	149.95	159.95
4 MODULE CABINET - unfinished	150.00	-----	-----
POWER SUPPLY w/disk protect	250.00	-----	-----




Color Computer

MONOLINK - 20 Mhz Monochrome video driver	15.00	20.00
CC30 PORT BUS w/power supply 5 88-30, 2 Cart	169.95	199.95
POWER BOX 6 switched outlets transient suppression	29.95	39.95
RS-232 3-switched ports for above	ADD +20.00	+25.00

Write for FREE Catalog
ADD \$3.00 S&H PER ORDER
WIS. ADD 5% SALES TAX



**11931 W. Bluemound Road
MILWAUKEE, WIS. 53226
(414) 257-0300**

68' MICRO JOURNAL ADVERTISERS INDEX

'68' MICRO JOURNAL	60,67
AAA CHICAGO COMPUTER CENTER	36,37
ACORN COMPUTER SYSTEMS	70
ADVANCED DIGITAL TECHNOLOGY	6
ANALOG MICRO SYSTEMS	69
ANDERSON COMPUTER CONSULTANTS	67
CLEARBROOK SOFTWARE GROUP INC.	71
COLOR MICRO JOURNAL	9
COMPILER EVALUATION SERVICES	47
COMPUTER EXCELLENCE	48
COMPUTER PUBLISHING INC.	5
COMPUTER SYSTEMS CENTER	48,59
COMPUTER SYSTEMS CONSULTANTS, INC.	46
D.P. JOHNSON	66
DATA-COMP	18C
DIGITAL RESEARCH COMPUTERS	62,63
DIPL. ING. JURGEN KNAUFT	68
F&D ASSOCIATES	64
FISHER SCIENTIFIC	1FC
FRANK HOGG LABORATORY, INC.	69
GIMIX, INC.	3,72
GREAT PLAINS COMPUTER CO.	66
HAZELWOOD COMPUTER SYSTEMS	08C
INTERFACING TECHNOLOGIES, CORP.	66
INTROL CORP.	49
JBM	56,67
JOTO ASSOCIATES	71
KADAK PRODUCTS LTD.	65
LSI ENTERPRISES LTD.	58,64
MACROPLEX SOFTWARE	68
MICROWARE SYSTEMS CORP.	1,4
ROBERTSON ELECTRONICS	60
SMOKE SIGNAL BROADCASTING	7
SOUTH EAST MEDIA	50,51,52,53,54,55
STAR-KITS	48
TALBOT MICROSYSTEMS	68
TECHNICAL SYSTEMS CONSULTANTS	57
TERMINUS DESIGN, INC.	69
THOMAS INSTRUMENTATION	71
UNITEK	60
UNIVERSAL DATA RESEARCH, INC.	8
WESTCHESTER APPLIED BUSINESS SYSTEMS	64
WINDRUSH MICRO SYSTEMS LIMITED	61

This Index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

THOMAS INSTRUMENTATION

HARDWARE

CPU-5
S-R/R w/o memory chips
S-R/R with 48K
SP-1
6802 SUPER CPU
VIDEO RAM
PARALLEL I/O
SS-50/50C EXTENDER
SS-30 EXTENDER
SS-50 WIRE WRAP
SS-30 WIRE WRAP
SS-30 BACKPLANE 8POS.
SS-50 BACKPLANE 4,8,12,
& 16 @ \$5.00/SLOT

ASSEM & TEST

\$299.00
\$120.00
\$399.00
\$195.00
\$235.00
\$195.00
\$139.00
\$ 35.00
\$ 25.00
NA
NA
NA
NA

BARE CARD

\$49.00
\$49.00
\$49.00
\$49.00
\$59.00
\$49.00
\$49.00
NA
NA
\$39.00
\$20.00
\$39.00
NA

SOFTWARE

VDISK
Use extended memory
as a fast disk drive
6809 source & obj \$149.00
6809 object \$ 99.00
OUTSIDE MODEM PROGRAM
includes source
UniFLEX™ \$100.00
FLEX™ 6800 or 6809 \$ 50.00
CROSS ASSEMBLER
For 6800, 6801, 6805
Runs on 6809 FLEX™
\$150.00

COMPONENTS

GOLD MALE MOLEX \$1.60 GOLD FEMALE MOLEX \$1.60
TIN MALE MOLEX \$.40 TIN FEMALE MOLEX \$.50
146805E2P \$20.00

THOMAS INSTRUMENTATION

168 EIGHTH STREET
AVALON, N.J. 08202
(609) 967-4280

MASTERCARD, VISA, AND C.O.D. ACCEPTED. PLEASE ALLOW 3 WEEKS
FOR CHECKS TO CLEAR. CONT. USA ADD \$3.00 SHIPPING. CANADA
\$6.00. FOREIGN \$12.00. NJ RESIDENTS ADD 6% SALES TAX
FLEX and UniFLEX ARE TRADEMARKS OF TECHNICAL SYSTEMS CONSULTANTS



BT9 (CSG - BT9) NEW IMPROVED ALGORITHM BT9

allows BASIC and C programmers to easily and quickly write programs in which data must be maintained in sorted order. Several indexes may be maintained for the same data file. BT9 now uses three structures for greatly improved speed and reliability. Written in C.

DISK UTILITIES PACKAGE (CSG - DUP)

Four utilities designed to perform commonly used functions for an entire disk. All four utilities operate on the directory specified and all directories in lower hierarchical levels in the file structure. Wildcard filename specification features are also supported. DDx, OCopy, DDel, DDir. Written in Machine Language.

DISK EDITOR (CSG - DEDIT) A screen oriented disk editor. DEDIT allows the user to view and modify any readable sector on an OS-9 format disk. Data is simultaneously displayed in hexadecimal, ASCII and binary formats. Modifications to a sector may be made by moving the cursor to the byte(s) to be changed, typing the correct data and issuing an Update command. A terminal with clear screen and cursor addressing functions is required. Written in Machine Language.

CSG - BT9 \$59.00
CSG - DUP \$49.00
CSG - DEDIT \$39.00

Specify 5 inch or 8 inch disk.
Prices include shipping and handling FOB
Surrey WA. Payment CERTIFIED check or
money order. VISA MC CDD.

CASH DISCOUNT Order two products
and deduct 5% from the total. Order three
products and deduct 10% from the total.
Discount applies to prepaid or credit
card orders only.

COMING SECOND QUARTER 1984

CSG-GL — a full featured General Ledger program for OS-9. Financial reporting system produces professional looking statements.

CSG-AR — a comprehensive Accounts Receivable program supporting both Balance Forward and Open-Item type accounts. Both programs use B-Tree files and advanced screen editing features.

FIRST CLASS

INTELLICOM

INTELLigent COMmunications Program

INTELLICOM provides you with the capability of intelligent computer to computer communications from both a terminal emulation and file transfer standpoint. INTELLICOM supports several file transfer protocols that facilitate the transfer of both binary and ASCII data. Since INTELLICOM is menu drive, it is a breeze to use and understand. With INTELLICOM you will be able to communicate with the various data and timesharing services such as The Source and CompuServe. Additionally, since INTELLICOM supports the protocol used by virtually all remote CP/M systems around the world, all users can immediately begin to take advantage of the wealth of public domain software available on these systems. Current, or potential, users of CompuServe can transmit and receive both binary and ASCII data with full error detection and recovery. The checksum protocols allow for the verifications of data blocks transferred (assuming appropriate support on the host end). This feature will be of great value in those applications where data integrity is paramount. INTELLICOM's documentation includes a detailed description of all protocols used along with machine readable examples of host pseudo code. A telephone directory and dialing facility compatible with Hayes Smartmodems is supported.

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microvare

Price: OS9 or FLEX Version \$29.95, add \$6.00 CDD
JOTO ASSOCIATES, 104 Lepage Dr., Southington, CT 06488

203-621-8070

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.

72

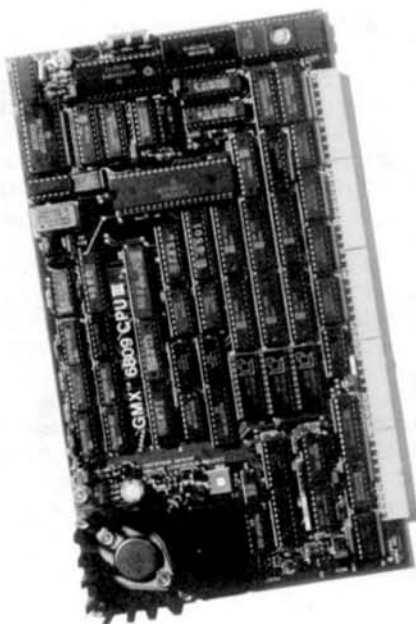
For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

GIMIX versions of OS9 and UniFLEX include maintenance and support by Micro-ware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

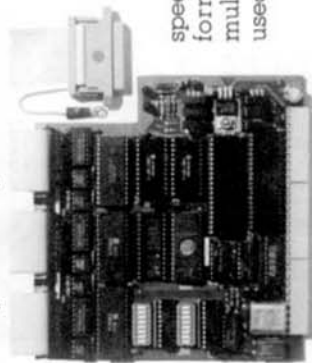
GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.



For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.

The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, there-

by freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.



GIMIX 6809 systems support five predominant operating systems:

**OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX**

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business; whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably

Please phone or write if you need further information.



GIMIX^{INC.}

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

© 1983 GIMIX Inc.

The Original FLEX™ for Color Computers

- Upgrade to 64K
- RS to FLEX, FLEX to RS file transfer ability
- Create your own character set
- Automatic recognition of single or double density and single or double sided
- All features available for either single or multiple drive systems
- Settable Disk Drive Seek Rates
- Faster High Resolution Video Display with 5 different formats
- Save RS Basic from RAM to Disk
- Move RS Basic to RAM
- Load and save function on FLEX disk
- 24 Support Commands 12 with Source Text
- External Terminal Program

DATA-COMP has everything you need to make your TRS-80C Color Computer WORK for YOU: from Parts and Pieces to Full Ready To Use Systems. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version + Editor & Assembler (which normally sell for \$50.00 ea.)	\$150.00
F-MATS(RS) FLEX9 Conversion Rout. for the RS Disk Controller when purchased with Special General FLEX9 Sys.	\$ 49.95
when purchased without the General FLEX9 Sys.	\$ 59.95
Color Computer with 64K RAM and EXT.BASIC	\$399.95

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9, F-MATS(RS), and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set.

\$999.95

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set.

\$799.95

Radio Shack Disk Controller	\$169.95
1 Single Sided, Double Density Disk Drive Tandem	\$229.95
1 Double Sided, Double Density Disk Drive Qume	\$319.95
1 Qume Thinline Double Sided, Double Density	\$249.95
1 Tandem 40 Track SSD with Cabinet and Power Supply	\$259.95
Single Drive Cabinet with Power Supply	\$ 79.95
Double Drive Cabinet with Power Supply	\$189.95

DISK DRIVE PACKAGES, etc.

These Packages Include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable:

PAK #1 - 1 Single Sided, Double Density Sys.	\$389.95
PAK #2 - 2 Single Sided, Double Density Sys.	\$675.95
PAK #3 - 1 Double Sided, Double Density Sys.	\$569.95
PAK #4 - 2 Double Sided, Double Density Sys.	\$919.95
PAK #5 - 2 Qume Thinline Double Sided, Double Density Sys.	\$699.95
PAK #6 - 2 Tandem Thinline Single Sided, Double Density Sys.	\$599.95

PRINTERS

EPSON RX-80	\$375.00
EPSON MX-100	\$725.00
EPSON MX-80	\$395.00
EPSON PX-80	\$599.00

SERIAL BOARDS

MX-Series	\$119.95
RX-FX-Series	\$ 99.95
Spectrum MX-Series	\$ 54.95

Mark Data Keyboards \$ 67.95

FLOPPY DISKETTES

MEMOREX VERBATIM

5" Soft Sector Disks		
Single Sided Single Density	\$2.60ea.	\$2.75ea
Single Sided Double Density	\$2.40ea	\$2.75ea
Double Sided Double Density		\$4.92ea
Plastic Storage Box		\$2.00ea
8" Soft Sector Disks		
Single Sided Single Density		\$3.75ea
Single Sided Double Density		\$4.10ea
Double Sided Double Density		\$4.75ea
Plastic Library Box		\$5.00ea

PARTS AND PIECES

Single Drive Disk Cable for RS Controller	\$ 19.95
Double Drive Disk Cable for RS Controller	\$ 24.95
Micro Tech. Prods. Inc. LOWER CASE ROM Adapter	\$ 74.95
Radio Shack BASIC Version 1.1 ROM or 1.2 ROM	\$ 34.95
Radio Shack Extended Basic ROM	\$ 74.95
Radio Shack Disk Basic ROM 1.1	\$ 29.95
Screen Clean - CL - rs Up Video Distortion On Your Color Computer	\$ 39.95
Set of Eight 64K RAM Chips w Mod Instructions	\$ 49.95

For Ordering Call TOLL FREE 1-800-338-6800

LAST OF PRODUCTION!!!

!! LIMITED QUANTITY !!

**SWTPC 8212 & 8212(W)
Intelligent Terminals**

New & Demo Models

At Discount Prices

**Remaining Supply of
SWTPC 8212 CRT Terminals**

**EM-82 Video
Terminal**

Emulates the 8200 Series from SWTPC



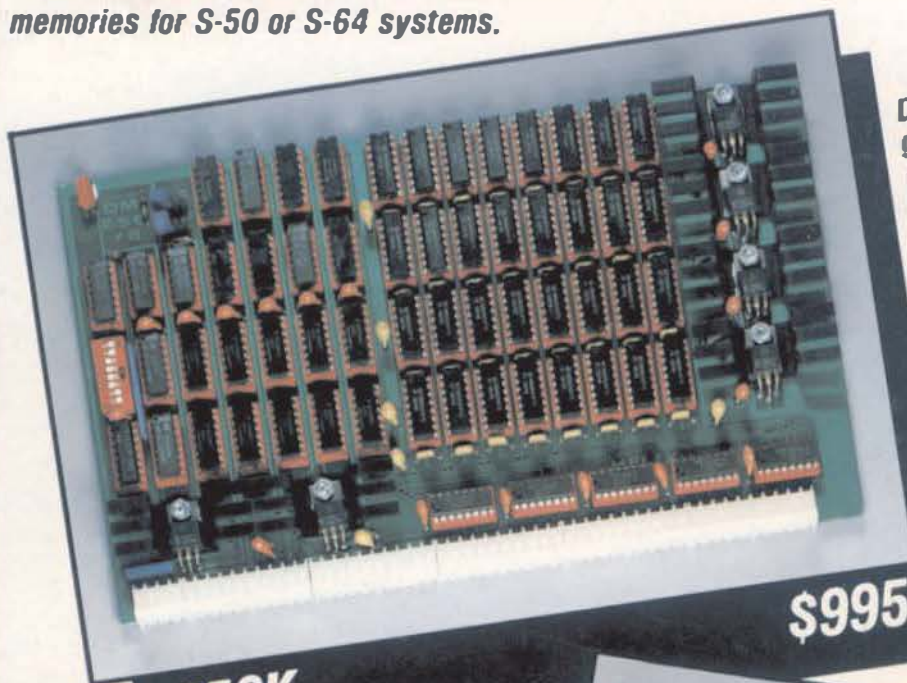
*FLEX is a trademark of Technical Systems Consultants
*OSB is a trademark of Microware

Data-Comp—South East Media & 68 Micro Journal Are Divisions of CPI

KINGSTON SPRINGS TN 37082
P.O. BOX 87
MR. MICKEY FERGUSON
000422 A/E

MEMORY • MEMORY • MEMORY • MEM

Hazelwood Computer Systems presents a pair of advanced memories for S-50 or S-64 systems.



256K

\$995

Dynamic memory with guaranteed 2.5 MHz operation

The DM-266 utilizes the same proven refresh method as our original DM-64. This technique allows fully transparent refresh while maintaining full bus speed and eliminating difficult timing problems.

In addition, individual power regulators for each row of memory ICs enhance reliability by providing the ultimate in noise decoupling. Flexible addressing mode controls make the entire 256K available to OS9 Level 2 or UniFLEX. For FLEX, the furnished utility program HYPERDSK™ formats memory not used by FLEX as a virtual, super fast disk, thus tremendously improving total system performance.

With several hundred already in use, the DM-266 is the best choice for economical, reliable memory.

Assembled, tested, and burned in
Order: DM-266



64K

\$395

Static memory with on-board battery backup

Utilizing the latest in low power CMOS static RAMs, the CM-64 has an on-board NiCad battery to maintain memory contents in the absence of bus power. Generous timing margins allow operation at bus speeds in excess of 2 MHz. Zener diode power loss sensing inhibits write operations to prevent data loss during power transitions. Write access can be explicitly controlled by means of an on-board jumper, a remote mounted switch, or an external logic signal.

This allows the memory to appear as ROM to the computer after the contents have been "programmed" using normal computer operations. The economical price makes the advantages of non-volatile memory available to any system.

Assembled, tested, and burned in
Order: CM-64

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366,

314-281-1055

HELIX